

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ
ТРАНСПОРТА (МИИТ)»**

**ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ
УПРАВЛЕНИЯ (ИТТСУ)**

Кафедра «Управление и защита информации»

А.И. САФРОНОВ, Н.Н. ЗОЛЬНИКОВА, В.Г. НОВИКОВ

**СОСТАВЛЕНИЕ ОТЧЁТНОЙ ДОКУМЕНТАЦИИ ПО РЕШЁННЫМ
ЗАДАЧАМ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ**

Учебно-методическое пособие

для проведения аудиторных занятий по Учебной практике

МОСКВА – 2018

МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ

ТРАНСПОРТА (МИИТ)»

ИНСТИТУТ ТРАНСПОРТНОЙ ТЕХНИКИ И СИСТЕМ

УПРАВЛЕНИЯ (ИТТСУ)

Кафедра «Управление и защита информации»

А.И. САФРОНОВ, Н.Н. ЗОЛЬНИКОВА, В.Г. НОВИКОВ

СОСТАВЛЕНИЕ ОТЧЁТНОЙ ДОКУМЕНТАЦИИ ПО РЕШЁННЫМ
ЗАДАЧАМ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие

для бакалавров направления

27.03.04 «Управление в технических системах»

МОСКВА – 2018

Сафронов А.И., Зольникова Н.Н., Новиков В.Г. Составление отчётной документации по решённым задачам алгоритмизации и программирования: Учебно-методическое пособие для проведения аудиторных занятий по Учебной практике. – М.: РУТ (МИИТ), 2018. – 83 с.

В учебно-методическом пособии рассмотрены основные идеи и принципы составления отчётной документации к решённым обучающимися РУТ (МИИТ) типовым задачам программирования и алгоритмизации. Материалы, собранные в учебно-методическом пособии, ориентированы на обучающихся, которые впервые приступили к изучению основ алгоритмизации и программирования. Издание содержит большое количество иллюстраций, способствующих изучению и закреплению на практике изложенных методов. Для контроля усвоения обучающимися материала в качестве основной задачи им предложена самостоятельная подготовка блок-схем алгоритмов к фрагментам ранее составленного, отлаженного и работоспособного программного обеспечения, что позволяет обучающимся развить навыки анализа кода программного обеспечения, а также видение однозначного соответствия блоков схем алгоритмов использованным в коде операторам языка *Visual C#*.

Рецензент:

Доцент кафедры «Радиофизика, антенны и микроволновая техника» Московского авиационного института (национального исследовательского университета), к.т.н. Ильин Е.В.

Введение

Работы, выдаваемые преподавателями студентам вузов, ориентированы не только на развитие профессиональных, но также и на оттачивание общекультурных компетенций. В числе общекультурных компетенций, как правило, значатся навыки составления обучающимися отчётной документации. Одной из учебных дисциплин, в рамках которой имеются ресурсы и возможности для эффективного и быстрого ознакомления обучающихся с методами и основами составления отчётной документации к задачам алгоритмизации и программирования, является Учебная практика – практика по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков научно-исследовательской деятельности.

На страницах данного учебно-методического пособия рассматриваются основные идеи и принципы составления отчётной документации к решённым обучающимися РУТ (МИИТ) типовым задачам программирования и алгоритмизации.

Издание нацелено на создание унифицированной методики подготовки обучающимися отчётной документации по всем видам работ, которые они выполняют на протяжении четырёх лет обучения по программе бакалавриата на кафедре «Управление и защита информации».

В учебно-методическом пособии ставится конкретная задача по составлению отчётной документации с акцентом внимания на детализации существующих правил оформления, в частности, правил оформления титульного листа, оглавления и библиографического списка. Правила оформления библиографического списка вынесены в Приложение, поскольку не являются обязательным пунктом выполнения поставленной в

учебно-методическом пособии задачи. Размещение столь важной составляющей отчётной документации в Приложении связано ещё и с тем обстоятельством, что рассматриваемая задача выдаётся обучающимся первого курса на первом же аудиторном занятии по Учебной практике. Объёмное задание в состоянии вызвать перегрузку и отказ обучающихся от его выполнения.

Анализ работ, выполняемых обучающимися по различным дисциплинам на кафедре «Управление и защита информации» показал, что за четыре года они не сталкиваются с такими их видами (за исключением выпускной квалификационной работы), в которых требовалось бы оформление обширного (состоящего более, чем из десяти позиций) библиографического списка. В таком списке правильные позиции источников становится проблематично отследить и оперативно скорректировать. Так вот, поскольку за четыре года существует только одна работа подобного масштаба, то автоматизация составления библиографического списка не является обязательной для выполнения в рамках самой первой задачи Учебной практики.

Раздел, посвященный оформлению автособираемого библиографического списка, содержится в учебно-методическом пособии исключительно как справочная информация, к которой обучающиеся могли бы в любой момент прибегнуть в случаях острой на то необходимости, например, в период написания выпускной квалификационной работы.

Для контроля усвоения обучающимися материала им предлагается к выполнению самостоятельная разработка блок-схем алгоритмов к фрагментам ранее составленного, отлаженного и работоспособного программного обеспечения,

что позволяет им развить навыки анализа кода программного обеспечения, а также видение однозначного соответствия блоков схем алгоритмов использованным в коде операторам языка *Visual C#* [1, 2].

1 Задание 1

Овладеть навыками составления отчётной документации на примере создания и наполнения отчёта к поставленной задаче программирования в рамках Учебной практики. В процессе выполнения задачи необходимо освоить методику составления титульного листа к типовой задаче программирования; проанализировать составленный на языке *Visual C#* фрагмент кода; научиться составлению блок-схем алгоритмов программ в графическом редакторе схем *Microsoft Office Visio* по имеющемуся в наличии исходному коду.

Представленная ниже методика позволяет приобрести навыки работы с инструментами разметки и форматирования, входящими в состав текстового редактора *Microsoft Office Word*.

По итогам выполнения работы у обучающихся должен получиться шаблон типового отчёта о решённой задаче Учебной практики. Здесь следует обратить внимание на то, что задачи Учебной практики подразделяются на вычислительные задачи и задачи программирования, потому структура отчётов к каждому из упомянутых типов задач будет незначительно отличаться. Отчёт составляется в текстовом редакторе *Microsoft Office Word*.

2 Цель работы

В данном разделе приводится одна из возможных формулировок цели работы. Итого, цель состоит в изучении обучающимися базовых функций, входящих в состав графического редактора для составления схем (в частности, блок-схем алгоритмов) *Microsoft Office Visio*, а также базовых функций текстового редактора *Microsoft Office Word*. Сформулированная цель работы способствует приобретению обучающимися навыков составления отчётной документации к решённым задачам программирования и алгоритмизации.

3 Структура отчёта

Отчёт по выполненной работе должен содержать следующие компоненты и разделы:

1. Титульный лист.
2. Оглавление (строго автособираемое).
3. Формулировку цели работы.
4. Описание задачи согласно выданному варианту.
5. Содержательную часть:
 - 5.1. Анализ фрагмента кода с текстовым описанием выявленных его особенностей.
 - 5.2. Составление фрагмента блок-схемы алгоритма по фрагменту кода.
6. Формулировку вывода о проделанной работе (вывод формулируется обезличено, то есть в нём должны

отсутствовать местоимения, такие как: «я», «мы» и другие).

Далее сформулирован перечень рекомендаций к отчёту, выполнение которых может стать весомым аргументом, доказывающим самостоятельность выполнения работы обучающимся, а также существенно упростить процедуру проверки преподавателем составленного отчёта:

1. Выполнение дополнительных иллюстраций / рисунков / скриншотов для демонстрации действий по изменению разметки «до» / «после» («было» / «стало»).
2. Подготовка нумерации иллюстраций / рисунков / скриншотов (если таковые имеются) с подписями, содержащими названия иллюстраций / рисунков / скриншотов, например, «Рисунок 1 – Набор стандартных фигур *Microsoft Office Visio*».

Внимание! Отчёт по выполненной работе сдаётся в электронном виде и на бумажном носителе. Электронный вид, как правило, направляется на адрес электронной почты преподавателя. Такое письмо должно быть снабжено либо файлом архива (*.rar, *.zip), либо двумя отдельными файлами: с отчётом *Microsoft Office Word* (*.docx) и с составленной блок-схемой алгоритма в *Microsoft Office Visio* (*.vsd, *.vsdx).

4 Требования к именам файлов:

Общий вид формата: «Дата. Задание. Фамилия.docx».

Формат даты: «ГГГГММДД», где ГГГГ – четыре цифры текущего года, ММ – две цифры текущего месяца, ДД – две цифры текущего дня.

Формат задания: «Задание NNk », где NN – две цифры номера задания, k – обозначение «о», если файл содержит общую часть, «и», если файл содержит индивидуальную часть, «ои», если файл содержит как общую, так и индивидуальную части.

Если устранить замечания по работе обучающемуся удаётся в тот же день: после фамилии ставится пробел и в круглых скобках записывается номер попытки исправления работы.

Если изменение имён файлов нарушает корректную работу программы: допускается упаковка всей совокупности имеющихся в наличии файлов в архив «*.rar» или «*.zip» с именем, соответствующим формату «Дата. Задание. Фамилия.rar»

Примеры правильных имён файлов, которые сдаются на проверку впервые:

«20180520. Задание 01о. Иванов.vsdх»;

«20180309. Задание 01ои. Иванов.docх»;

Примеры правильных имён файлов, которые сдаются на проверку повторно в тот же день:

«20180520. Задание 01о. Иванов (1).vsdх»;

«20180309. Задание 01ои. Иванов (3).docх».

5 Методика оформления титульного листа:

Важно! Помните, что выбранное для титульного листа семейство шрифта (например, «*Times New Roman*») должно использоваться на протяжении всего документа. Семейство

шрифта должно быть выбрано любое строгое на усмотрение обучающегося.

К строгим семействам шрифтов относят: «*Times New Roman*», «*Tahoma*», «*Verdana*», «*Calibri*» и другие. К нестрогим семействам шрифтов: «*Majestic*», «*Lucida*», «*Comic Sans MS*», «*Gill*», «*Decor*», «*Calligraph*» и другие.

Размер шрифта рекомендуется выбрать величиной в 16 пунктов (16 пт). Следует помнить, что некоторые разделы титульного листа должны принципиально отличаться по величине шрифта от остальных разделов, что будет отдельно оговорено далее по тексту.

Титульный лист состоит из следующих разделов (Рисунок 5.1):

- ведомственная принадлежность вуза;
- регалии вуза;
- наименование вуза с указанием его сокращённого наименования;
- наименование института (с указанием сокращённого наименования или без него);
- наименование кафедры, к которой относится дисциплина (с указанием сокращённого наименования или без него);
- вид документа;
- вид работы (раздел присутствует не во всех документах);
- наименование дисциплины;

- тематика;
- «штампы» выполнения / проверки;
- строка «Город – год».

Министерство транспорта Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования

«Российский университет транспорта (МИИТ)» (РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Отчёт

по практике
«Учебная практика»

(вид практики)

Практика по получению первичных профессиональных умений
и навыков, в том числе умений и навыков научно-
исследовательской деятельности

(наименование практики)

Представлено решение задачи №1
«Составление отчётной документации»

Выполнил: ст. гр. ТУУ-111
Иванов И.И.
Вариант №1
Проверил: доц. Сафронов А.И.

Москва – 2018 г.

Рисунок 5.1 – Пример оформления титульного листа к отчёту по учебной практике в *Microsoft Office Word*

В 2018 году РУТ (МИИТ) ведомственно принадлежит к Министерству транспорта Российской Федерации и является Федеральным бюджетным образовательным учреждением высшего профессионального образования. Полное наименование вуза следующее: «Российский университет транспорта (МИИТ)». Рядом записывается сокращённое наименование в круглых скобках «(РУТ (МИИТ))». Обратите внимание – все скобки расставлены верно. Одна закрывающая скобка здесь не является ошибкой или опечаткой.

Работа выполняется на кафедре «Управление и защита информации» («УиЗИ»), которая относится к Институту транспортной техники и систем управления (ИТТСУ).

Вид документа в рассматриваемом случае – «Отчёт», в иных случаях здесь может записываться «Курсовая работа», «Курсовой проект», «Реферат», «Эссе» и так далее. Обратите внимание, что надпись с наименованием вида документа по величине шрифта должна быть на 5 ступеней больше (не пунктов, а именно ступеней, которые соответствуют строкам в выпадающем списке возможных размеров шрифта), чем основной выбранный размер шрифта. Например, если основной размер «16 пт», то для вида документа он должен составлять «26 пт».

В отдельных отчётных документах ниже под надписью с наименованием вида документа должно прописываться наименование вида работы, например, «по практическому заданию», «по лабораторной работе» и тому подобное. В случае с отчётом по учебной практике вид работы не указывается.

Далее на титульном листе прописывается вид дисциплины. Как правило, большинство дисциплин имеют простые наименования, такие как «Физика», «Алгоритмизация и

технологии программирования», потому в соответствующей строке прописывается следующий текст: "по дисциплине: «Физика»".

Всем практикам согласно учебному плану присвоено составное название, потому в отчётной документации строго прописывается вид практики (в данном случае: «Учебная практика») и далее наименование этой практики (в данном случае: «Практика по получению первичных профессиональных умений и навыков, в том числе умений и навыков научно-исследовательской деятельности»).

Тематика. Для курсовых проектов, лабораторных работ и практических задач тематика определяется по наименованиям этих работ, записанных в методических указаниях или сформулированных преподавателем в устной форме. Каждая из задач учебной практики различается по тематике, потому на титульном листе отчёта к каждой из задач учебной практики прописывается её наименование, согласно постановке. Так тематикой данной работы является «Составление отчётной документации». Согласно личным требованиям руководителя практики в рассматриваемом случае строка, содержащая тематику работы, полностью должна выглядеть следующим образом: "Представлено решение задачи №1 «Составление отчётной документации»".

Ниже следует «штамп» выполнения / проверки работы. Под «штампом» понимается текст типовой структуры (Рисунок 5.2).

Выполнил: ст. гр. ТУУ-111
Иванов И.И.
Вариант №1
Проверил: доц. Сафронов А.И.

Рисунок 5.2 – Пример оформления «штампа» выполнения / проверки работы

Размер шрифта «штампа» должен быть на один шаг меньше размера основного шрифта. Для аккуратного отступа и выравнивания текстовых элементов штампа рекомендуется выставлять левый ползунок отступа ближе к правому ползунку отступа, после середины листа (Рисунок 5.3).



Рисунок 5.3 – Изменение положения левого ползунка отступа для форматирования «штампа» выполнения / проверки работы

Завершает титульный лист строка «город – год», выравненная по центру. В ней прописывается город, в котором расположен вуз (в рассматриваемом случае – Москва), далее после постановки тире прописывается год, в котором сдаётся и подписывается работа. Работа может быть выдана преподавателем в одном году, но сдаваться обучающимся уже в следующем году (такая ситуация не редкость для задач, выдаваемых в первых учебных семестрах). В упомянутом случае обучающийся должен указывать именно год сдачи работы, а не год выдачи задания преподавателем.

По окончании форматирования титульного листа обучающийся должен выставить курсор в конце строки «город – год» и нажать сочетание клавиш «Ctrl» + «Enter», благодаря которому создаётся разрыв страниц. Курсор при этом

перемещается на следующую страницу, что соответствует началу работы в новом разделе. Этот шаг позволяет своевременно выявить ситуации перехода строки «город – год» с титульного листа на следующую страницу. В этом случае не потребуется перепечатывать весь документ целиком, – структура и наполнение его разделов сохранится. Нужно будет только поправить титульный лист и перепечатать его.

6 Методика настройки нумерации страниц

Нумерация страниц важна для удобства обсуждения документа с преподавателем, а также для удобства навигации по документу. Нумерация страниц делает документ более строгим и аккуратным. В *Microsoft Office Word* нумерация страниц выполняется автоматически в выбранных в настройках областях колонтитулов после выполнения нескольких простых шагов изложенной ниже методики.

Для реализации автоматической нумерации страниц в документе без учёта титульного листа необходимо перейти в меню «Вставка», где в разделе «Колонтитулы» найти пункт «Номер страницы» (Рисунок 6.1).

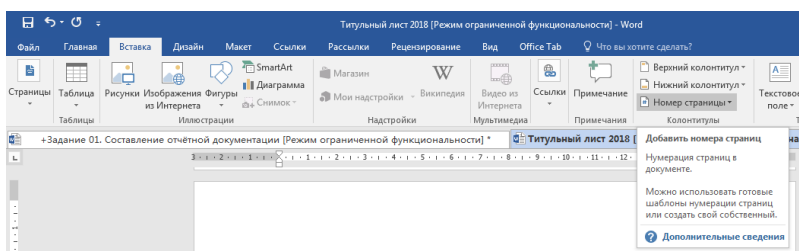


Рисунок 6.1 – Переход к настройке нумерации страниц

Как правило, в большинстве известных документов используется схема постановки номера страницы внизу и справа (Рисунок 6.2). Выберем эту схему.

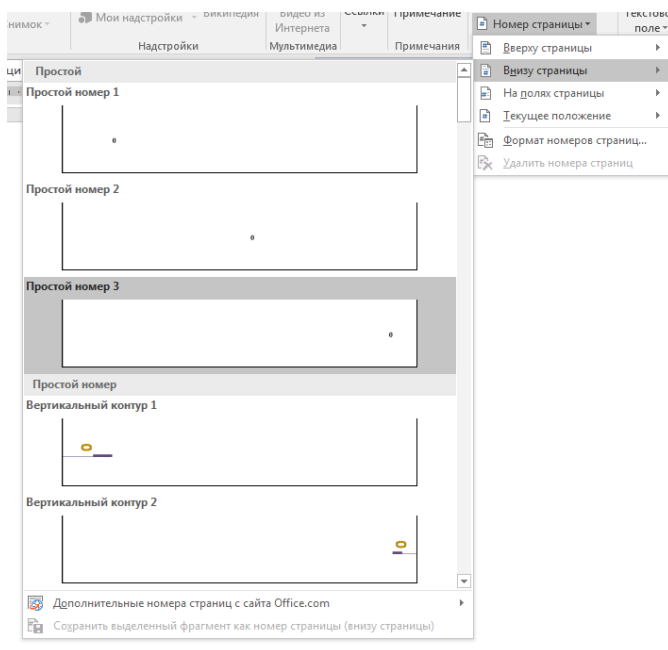


Рисунок 6.2 – Выбор вида нумерации согласно шаблону «снизу и справа», именуемому «Простой номер 3»

Выбор схемы характеризуется автоматическим переходом к настройкам колонтитулов. В меню *Microsoft Office Word* появляется вкладка «Конструктор» и пользователю предоставляется возможность внесения изменений в колонтитул (Рисунок 6.3).

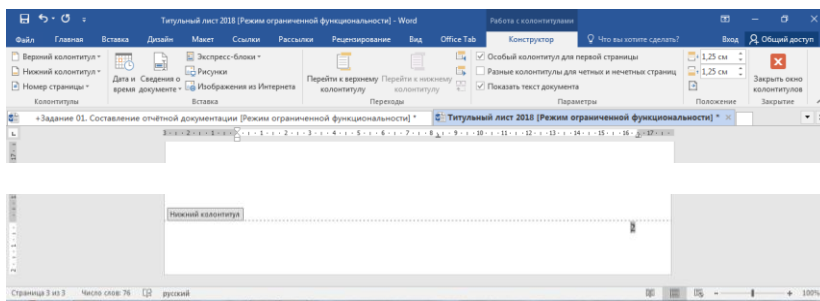


Рисунок 6.3 – Автоматический переход к настройке нижнего колонтитула; активация меню «Конструктор»

Обратите внимание, что для составления титульного листа могут потребоваться специфические настройки, например, отсутствие номера страницы. В этом случае важно проследить за тем, чтобы галочка напротив настройки «Особый колонтитул для первой страницы» была выставлена (Рисунок 6.4).

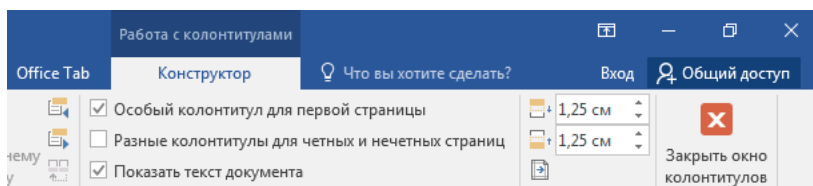


Рисунок 6.4 – Настройка особого колонтитула для первой страницы – титульный лист без номера страницы

На Рисунке 6.2 видно, что в подменю «Номер страницы» имеется пункт «Формат номеров страниц...», осуществляющий переход к одноимённому диалоговому окну с детализацией настроек номеров страниц (Рисунок 6.5).

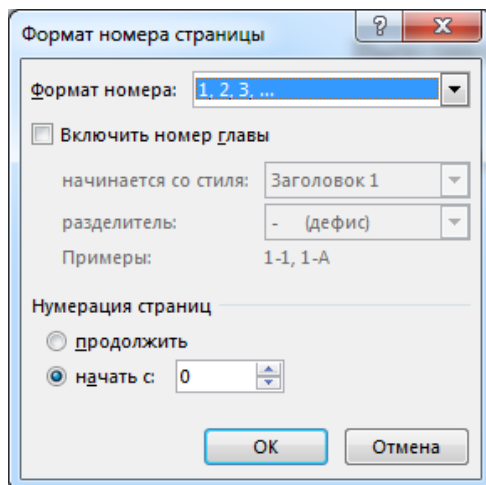


Рисунок 6.5 – Режим детальных настроек, при которых титульный лист не нумеруется и является нулевой страницей, а следующая за ним страница существует под номером «1», который отображается в колонтитуле

Представленная методика позволяет быстро настроить нумерацию страниц в документе *Microsoft Office Word*, однако, следует помнить, что некоторые похожие действия необходимо проделывать при нумерации страниц в документах с листами комбинированной ориентации (присутствуют как альбомные, так и книжные листы).

7 Методика настройки автособираемого оглавления

На следующей странице после титульного листа в отчётных документах формируется оглавление – перечень разделов и подразделов, входящих в состав документа, с указанием справа нумерации страниц, на которых эти разделы и подразделы располагаются. Наиболее удобным способом для

автоматического наполнения раздела является настройка автособираемого оглавления, реализуемая согласно методике, представленной далее по тексту.

Для начала работы с упомянутой настройкой необходимо перейти на новую, чистую страницу. Курсор выставляется в самом начале этой страницы, идущей вслед за титульным листом. Далее следует перейти в меню «Ссылки», раскрыть раздел «Оглавление» [3] и выбрать в нём пункт «Автособираемое оглавление 1» (Рисунок 7.1).

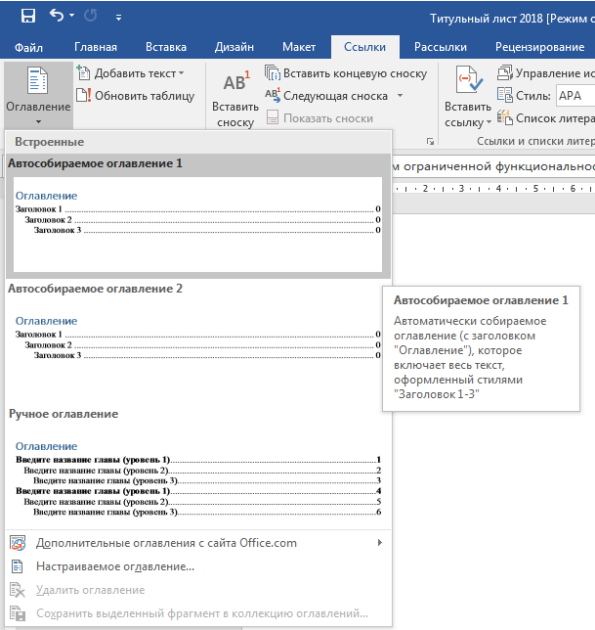


Рисунок 7.1 – Активация блока с автособираемым оглавлением в документе

После выбора указанного блока на листе появится следующая динамическая структура (Рисунок 7.2).

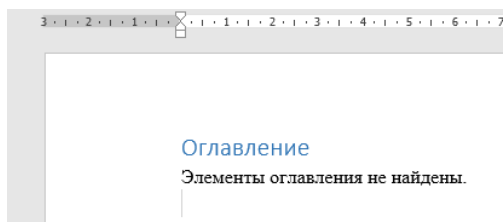


Рисунок 7.2 – Состояние автособираемого оглавления при отсутствии в документе какой-либо предварительно настроенной структуры заголовков и подзаголовков

Теперь, если выставить курсор на надписи «Оглавление» и далее перейти в главное меню, можно заметить, что в перечне стилей эта надпись определена как «Заголовок оглавления» (Рисунок 7.3).

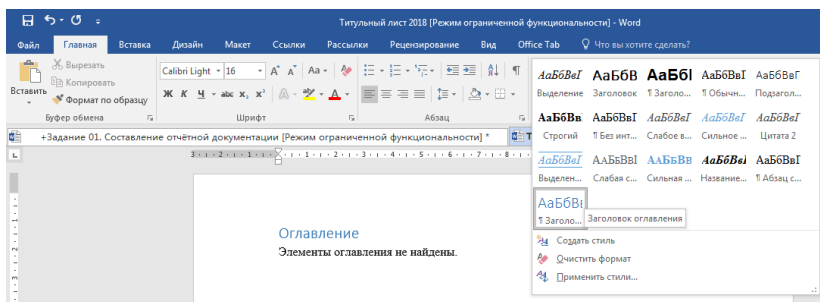


Рисунок 7.3 – Просмотр соответствия стиля надписи «Оглавление»

В серьёзных документах этот заголовок не должен быть синего или какого-либо иного цвета, отличного от чёрного, потому рекомендуется перенастроить умолчания *Microsoft Office Word* для всех последующих создаваемых на рабочем компьютере документов. Щелчок правой кнопкой мыши по области интересующего стиля инициирует вызов контекстного

меню этого стиля. В контекстном меню необходимо выбрать пункт «Изменить...» для перехода к деталям настройки стилового оформления текста (Рисунок 7.4).

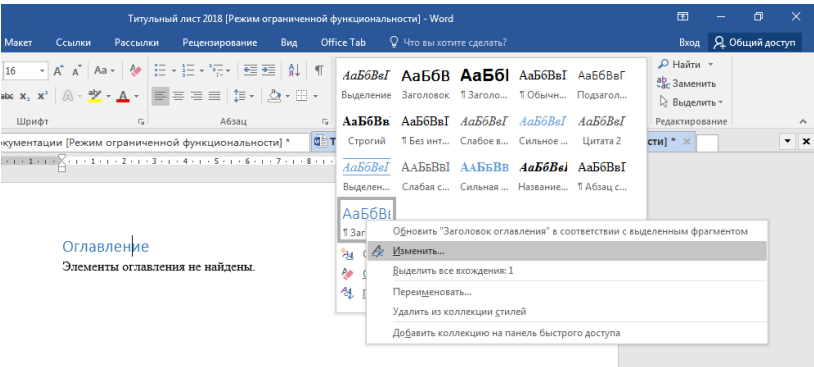


Рисунок 7.4 – Переход к диалоговому окну для изменения стиля текста

После выбора пункта «Изменить...» раскрывается диалоговое окно, в котором представлены текущие настройки выбранного стиля (Рисунок 7.5).

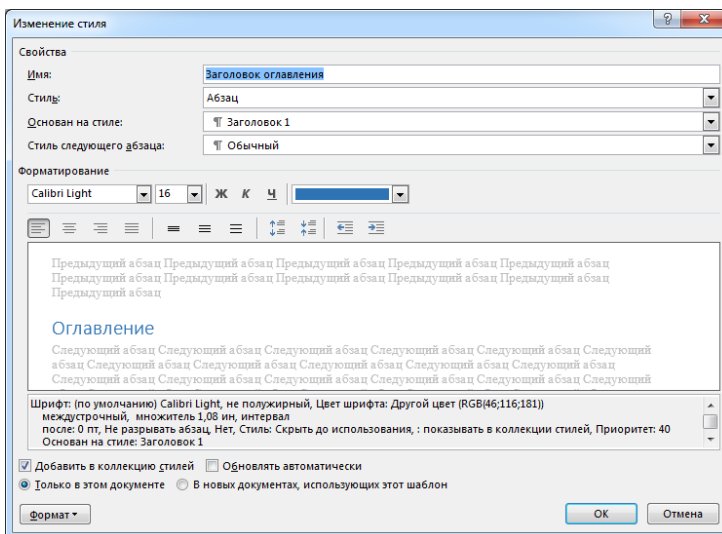


Рисунок 7.5 – Исходные настройки стиля «Заголовок оглавления»

Для выполнения отчётов к решённым задачам Учебной практики рекомендуется изменить эти параметры так, как показано на Рисунке 7.6. Следует обратить внимание на наличие, либо отсутствие галочки напротив настройки «Обновлять автоматически». Галка должна присутствовать. Также нижняя опция (круглый элемент), которая, как правило, по умолчанию расположена на позиции «Только в этом документе» должна быть перемещена на позицию «В новых документах, использующих этот шаблон». Последнее руководящее воздействие является факультативным и выполняется на усмотрение владельца персонального компьютера.

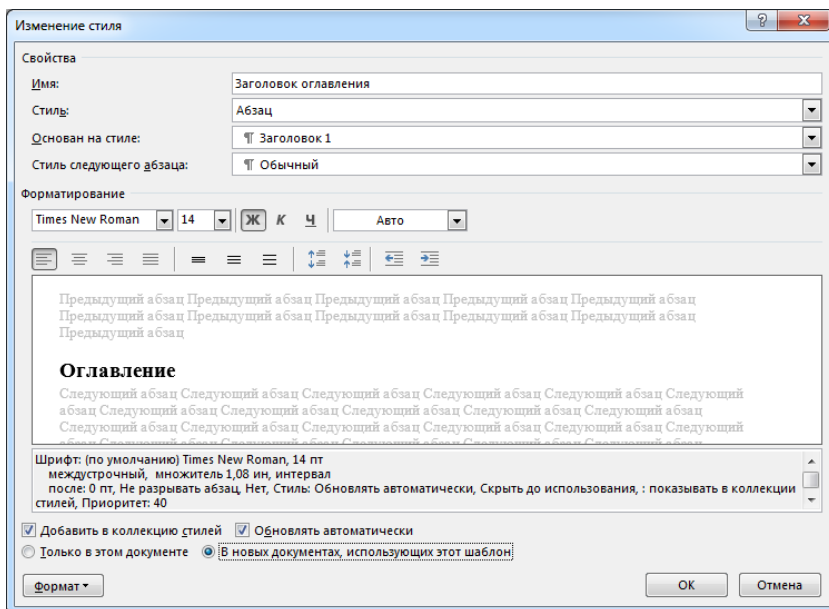


Рисунок 7.6 – Рекомендуемые настройки стиля «Заголовок оглавления»

После того как настройки вступят в силу, установленный ранее блок формирования оглавления примет вид, аналогичный представленному на Рисунке 7.7 виду.

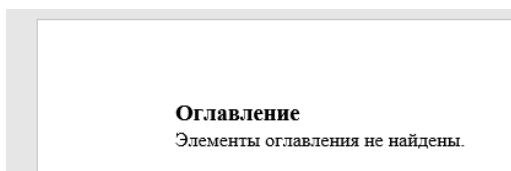


Рисунок 7.7 – Изменённое состояние блока автособираемого оглавления

Дальнейшим шагом, согласно рассматриваемой методике, необходимо перейти на следующую страницу с обязательным выполнением разрыва посредством нажатия сочетания клавиш «*Ctrl*» + «*Enter*».

На новой странице для примера запишем планируемые разделы будущего документа и внутри каждого дадим краткое описание планируемому наполнению этих разделов (Рисунок 7.8).

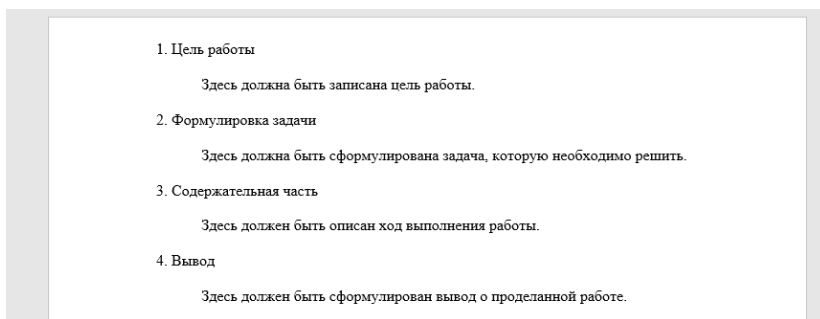


Рисунок 7.8 – Заготовка отчётного документа

Выделим строку, содержащую название первого раздела отчёта под названием «1. Цель работы» так, как это показано на Рисунке 7.9. Отчётливо видно, что названию записанного ранее раздела соответствует стиль «Обычный», заданный как умолчание для любого вводимого в документ *Microsoft Office Word* текста.

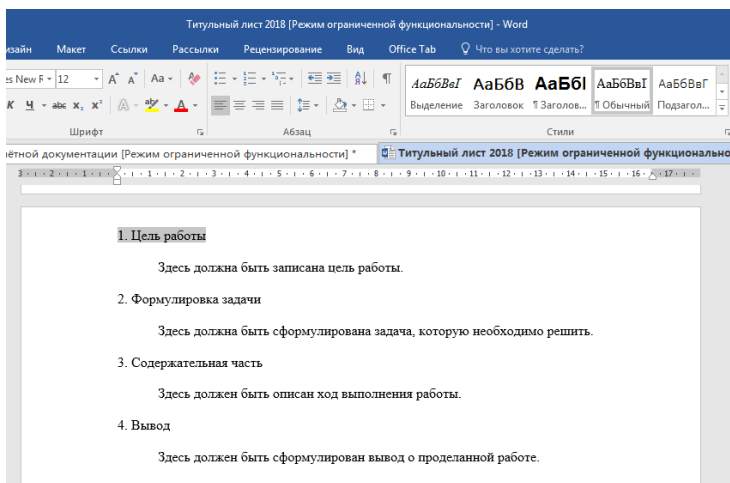


Рисунок 7.9 – Просмотр стиля типового фрагмента текста

Поменяем стиль выделенного фрагмента с «Обычного» на «Заголовок». Выделенный фрагмент автоматически преобразуется в соответствии с настройками, характерными для данного стиля (Рисунок 7.10).

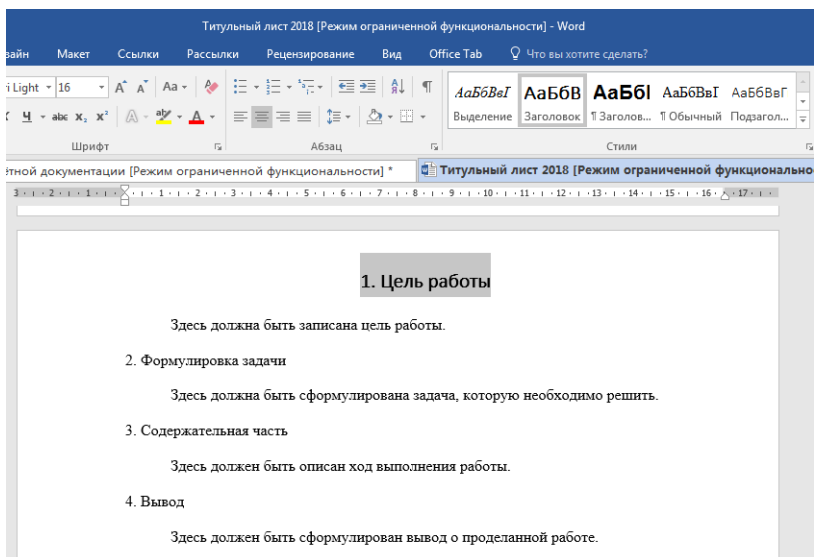


Рисунок 7.10 – Изменение стиля для выделенного фрагмента текста

После этого проверим, участвует ли шаблон стиля «Заголовок» в формировании автособираемого оглавления. Для этого вернёмся на предыдущую страницу составляемого документа и перейдём в меню «Ссылки», где выберем пункт «Обновить таблицу» (Рисунок 7.11). Если таблица обновляется впервые, то никаких дополнительных вопросов от текстового редактора *Microsoft Office Word* не последует.

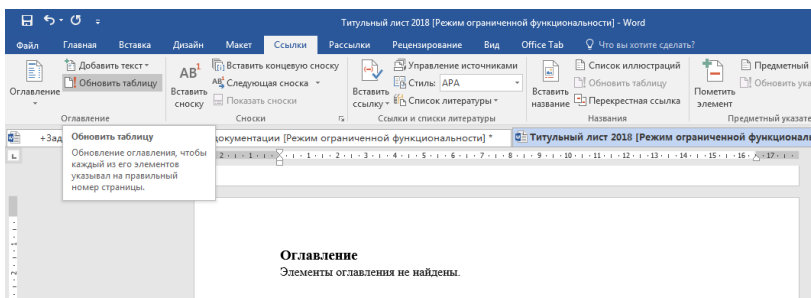


Рисунок 7.11 – Переход к обновлению таблицы с автособираемым оглавлением

В том случае, если применённый стиль оформления «Заголовок» участвует в формировании автособираемого оглавления, можно увидеть результат, аналогичный представленному на Рисунке 7.12.

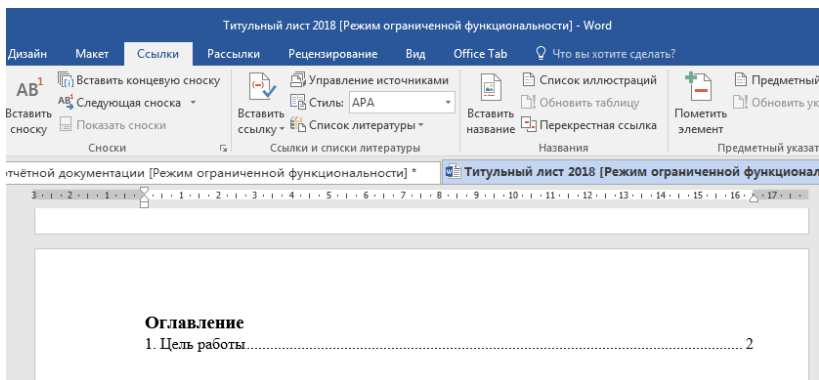


Рисунок 7.12 – Результат обновления таблицы с автособираемым оглавлением

Настроим по аналогии с «Целью работы» и остальные заголовки создаваемого шаблона отчётного документа так, как это показано на Рисунке 7.13.

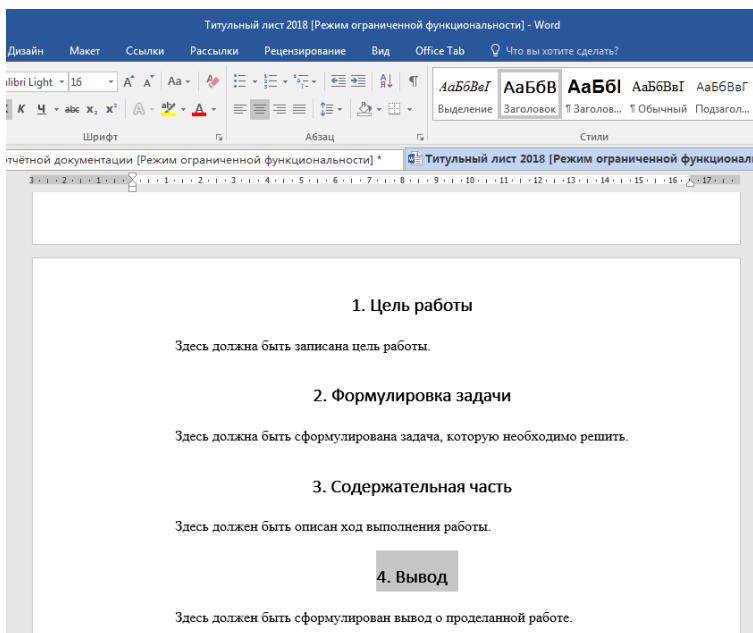


Рисунок 7.13 – Настройка всех планируемых заголовков под соответствующий стиль

Предположим, что заданные по умолчанию параметры: центрирование и размер шрифта заголовков нас не устраивают. В таком случае следует изменить эти умолчания, в том числе, и для стиля оформления заголовка по аналогии с тем, как это было проделано ранее с заголовком оглавления (Рисунок 7.14).

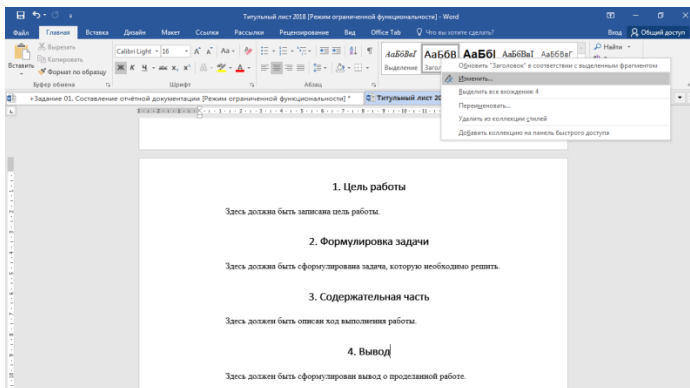


Рисунок 7.14 – Переход к изменению стиля «Заголовок» для текста

Далее на Рисунке 7.15 показаны настройки стиля «Заголовок», заданные в текстовом редакторе по умолчанию.

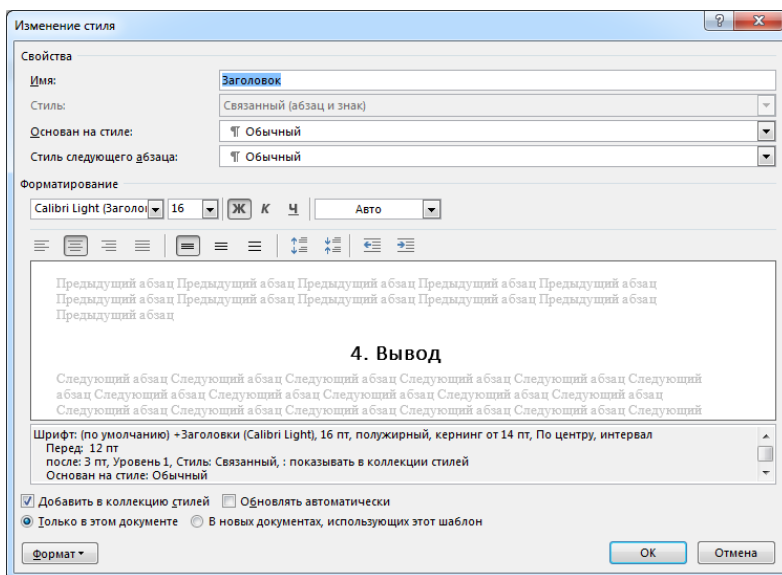


Рисунок 7.15 – Стилизовое умолчание для «Заголовков»

На Рисунке 7.16 представлены рекомендуемые настройки заголовков для *Word*-документов, содержащих отчёты по выполненным работам Учебной практики.

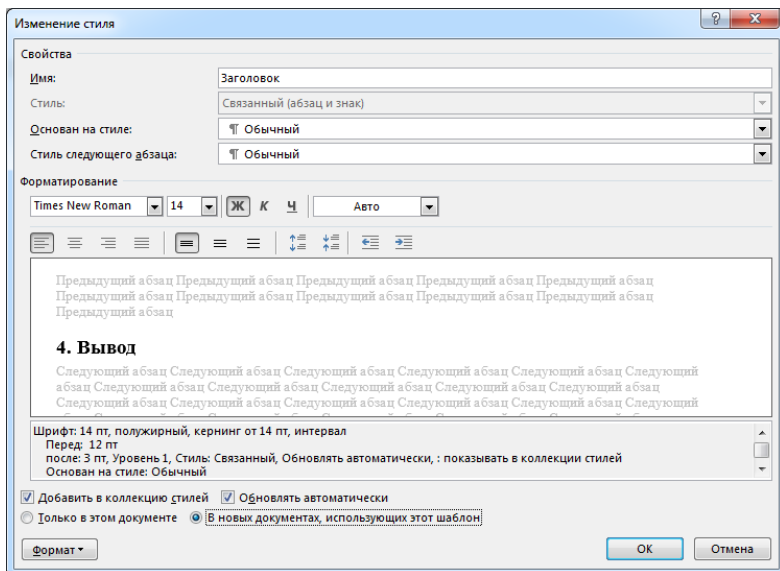


Рисунок 7.16 – Рекомендуемые настройки для стиля «Заголовок»

В результате применения настроек получаем документ, оформленный следующим образом (Рисунок 7.17).

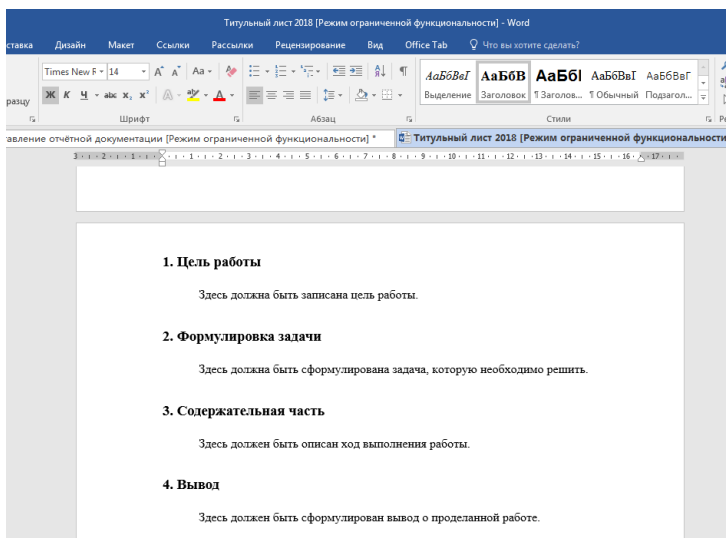


Рисунок 7.17 – Результат настройки заголовков в документе

После этого снова необходимо вернуться на предыдущую страницу, содержащую оглавление, перейти в меню «Ссылки» и выбрать пункт «Обновить таблицу». Обновление таблицы выполняется не впервые, потому вслед за выданной командой на исполнение операции последует диалог, в котором пользователю предлагается либо только обновить нумерацию страниц в имеющейся структуре, либо полностью обновить структуру таблицы. Полное обновление структуры предполагает дополнение существующего оглавления новыми компонентами (если таковые появились), а также изменение форматирования стилевого оформления (если стилиевые настройки менялись). Таким образом, при внесении каких-либо крупных изменений в настройки документа, в появившемся диалоговом окне (Рисунке 7.18) рекомендуется выбирать пункт «обновить целиком». Его и выбираем, подтвердив намерение нажатием на кнопку «ОК».

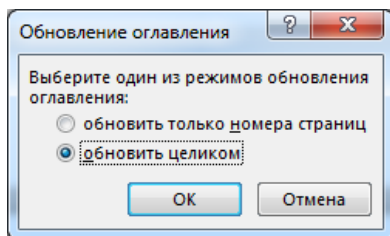


Рисунок 7.18 – Диалог-запрос на предмет необходимости обновления таблицы с автособираемым оглавлением

Если все действия согласно представленной методике были выполнены верно, то полученный результат компоновки автособираемого оглавления будет аналогичен представленному на Рисунке 7.19

Оглавление	
1. Цель работы.....	2
2. Формулировка задачи.....	2
3. Содержательная часть.....	2
4. Вывод.....	2

Рисунок 7.19 – Конечный результат формирования автособираемого оглавления

8 Методика составления блок-схем алгоритмов программ

Алгоритм – упорядоченная последовательность действий, приводящих к решению задачи [4].

Блок-схема алгоритма – это унифицированная пошаговая схема представления способа решения задачи программирования различной степени детализации [5].

В этом разделе представлено описание работы с графическим редактором схем *Microsoft Office Visio*. Упомянутое программное обеспечение необходимо для составления различного рода векторных схем, в частности, векторных блок-схем алгоритма. Обучающимся, в качестве самостоятельной работы, рекомендуется почитать в сети Интернет источники, в которых подробно описывается различие между растровой и векторной графиками.

Для начала работы по составлению блок-схем алгоритмов в *Microsoft Office Visio* необходимо запустить упомянутое программное обеспечение и из представленного перечня шаблонов документов выбрать вид «Простая блок-схема». Далее в вертикальной панели, расположенной слева, выбрать пункт «Дополнительные фигуры», далее «Блок-схема», далее «Фигуры простой блок-схемы». Выполнение перечня этих действий обусловит изменение вида вертикальной панели инструментов, расположенной слева. Вид станет аналогичным представленному на Рисунке 8.1.

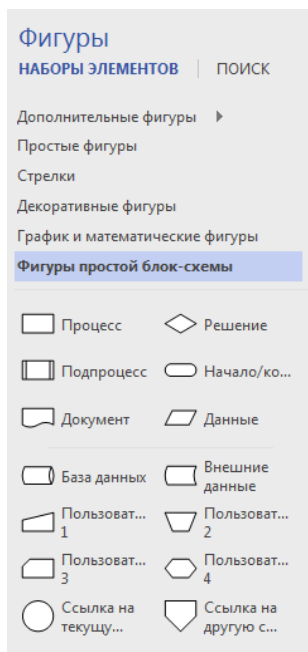


Рисунок 8.1 – Перечень фигур простой блок-схемы в *Microsoft Office Visio*

Далее рассмотрим типовые блоки, входящие в состав блок-схем алгоритмов программ, а также типовые блочные структуры, позволяющие проводить обучающимся аналогии с операторами, записанными в исходных кодах программ, составленных на алгоритмических языках программирования.

Программы, процедуры и функции, которые необходимо описывать полностью (не фрагментом), должны в начале и в конце содержать терминаторы (ограничители действий), именуемые окончательными фигурами (Рисунок 8.2).

Оконечные фигуры содержат отметки о том, начинается или заканчивается программа, а также обязательное наименование

программы / процедуры / функции. Распознать момент постановки начальной оконечной фигуры по коду можно благодаря служебным словам: «*public*», «*private*», «*static*», «*void*». Распознать момент постановки завершающей оконечной фигуры можно по встретившейся закрывающей операторной скобке «*}*», вслед за которой идёт одно из ранее упомянутых служебных слов («*public*», «*private*», «*static*», «*void*») или же отсутствие какого-либо текста.

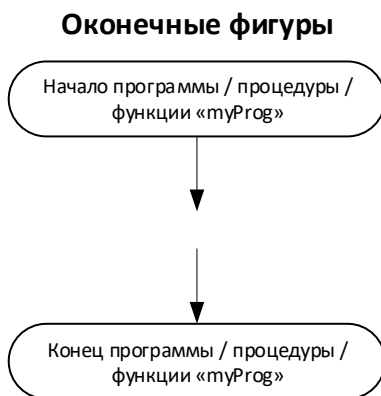


Рисунок 8.2 – Оконечные фигуры блок-схемы алгоритма программы

Блок, содержащий перечень переменных, входящих в состав программы, принято обозначать параллелограммом (Рисунок 8.3). Распознать переменную можно по имени, которому в коде предшествует тот или иной тип данных: «*int*» (целый), «*bool*» (логический), «*string*» (строковый), «*double*» (вещественный двойной точности), «*float*» (вещественный одинарной точности) и так далее. Ещё одним признаком является метод чтения консольного ввода с клавиатуры *Console.ReadLine()*.

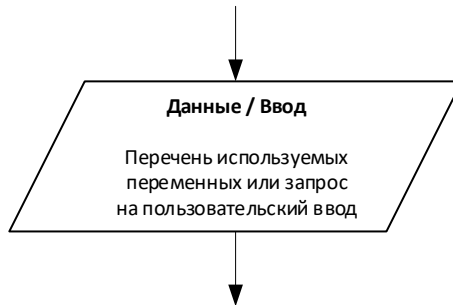


Рисунок 8.3 – Блок исходных данных и используемых в программе переменных

Любая операция, которая записывается в строке и заканчивается в коде постановкой разделителя операций «;», оформляется в блок-схеме алгоритма прямоугольным блоком, характеризующим тот или иной процесс (Рисунок 8.4).

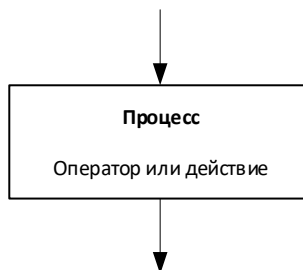


Рисунок 8.4 – Блок, содержащий оператор или группу операторов, разделённых между собой «;»

Интерфейсные элементы, не связанные со вводом информации, а демонстрирующие пользователю на экране или бумажном носителе конечный результат (выходные данные), оформляются блоком в форме оборванного бумажного листа

(Рисунок 8.5). Как правило, распознать операторы вывода информации можно по методам *MessageBox.Show()* или *Console.WriteLine()*.

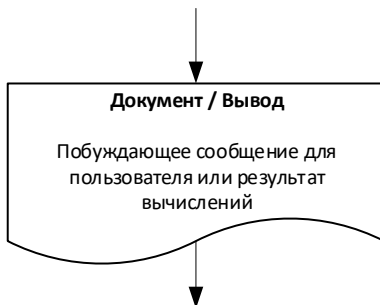


Рисунок 8.5 – Блок вывода результата

Распознать условный оператор (Рисунок 8.6), для которого характерен ромб на блок-схеме (так называемый блок решения или принятия решения), можно по служебному слову «*if*» (если). Оператор или группа операторов, расположенных вслед за конструкцией «*if*» в сочетании с логическим выражением, иллюстрируются на блок-схеме по направлению ветки «да» / «+». В том случае, если в исходном коде за упомянутым оператором или группой операторов указано ещё и служебное слово «*else*» (иначе), вслед за которым тоже записаны оператор или группа операторов, то на блок-схеме такие операторы иллюстрируются по направлению ветки «нет» / «-».

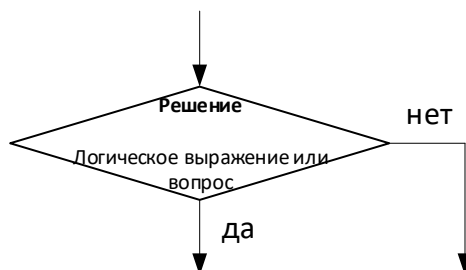


Рисунок 8.6 – Блок решения для обозначения условного оператора и входящих в его состав параметров

В тех случаях, когда встречаются служебные слова «*switch*» и «*case*», имеет место переключение / оператор переключения (Рисунок 8.7). Переключение характерно для ситуаций, в которых имеют место более, чем два состояния выходного сигнала. В качестве простого примера очень наглядно привести: сигналы светофора, месяцы в году, релейное поведение характеристики $(-1, 0, 1)$, наличие третьего состояния неопределённости (*true, false, null*).

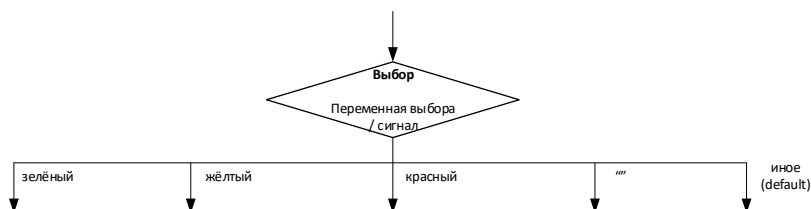


Рисунок 8.7 – Блок оператора выбора / переключения и входящих в его состав параметров

В том случае, когда наперёд известно, что множество объектов или значений должно обрабатываться по единому

сценарию, например, перебирается ли список сотрудников, которым должна быть назначена премия в случае, если весь месяц они приходили вовремя, или, например, известен диапазон значений, для которого необходимо построить график функциональной зависимости или просто записать значения функции в таблицу, перебрав все значения известного диапазона с некоторым шагом, речь идёт о повторе одного и того же участка кода многократно. Действие или действия, выполняемые по одинаковой схеме многократно для различных значений, группируются в теле циклического оператора, соответственно, в алгоритме имеет место циклическая конструкция. Когда количество повторов, то есть итераций цикла, заранее известно, то имеет место циклическая конструкция по известному диапазону значений (Рисунок 8.8). Распознаётся такая циклическая конструкция по наличию в коде операторов *«for»* и *«foreach»*.



Рисунок 8.8 – Блоковая структура цикла по известному
диапазону значений

Существуют и другие циклические конструкции. В тех случаях, когда диапазон перебираемых значений заранее неизвестен, но известно определяющее условие, которое прервёт выполнение однотипных действий в определённый момент времени, применяются циклические конструкции либо с предусловием (Рисунок 8.9), либо с постусловием (Рисунок 8.10). Для каждой из этих конструкций характерно наличие в исходном коде служебного слова «*while*». В первом случае «*while*» встречается в начале и обрамляет оператор или группу операторов операторными скобками «{», «}»; во втором случае циклическая конструкция открывается служебным словом «*do*» и после указания оператора или группы операторов завершается служебным словом «*while*».

В первом случае однотипные действия можно выполнять тогда и только тогда, когда параметры соответствуют предъявляемым к ним требованиям, иначе они ни разу не будут выполнены. Во втором случае оператор или группа действий выполняются сразу и только по результату их выполнения становится понятно, нужно ли повторить расчёт или всё сложилось удачно с первого раза и можно продвигаться по вычислительному тракту дальше. Во втором случае операторы, расположенные в теле цикла, хотя бы один раз, но обязательно выполняются, в первом случае такие операторы могут остаться не выполненными ни разу.



Рисунок 8.9 – Блочная структура цикла с предусловием

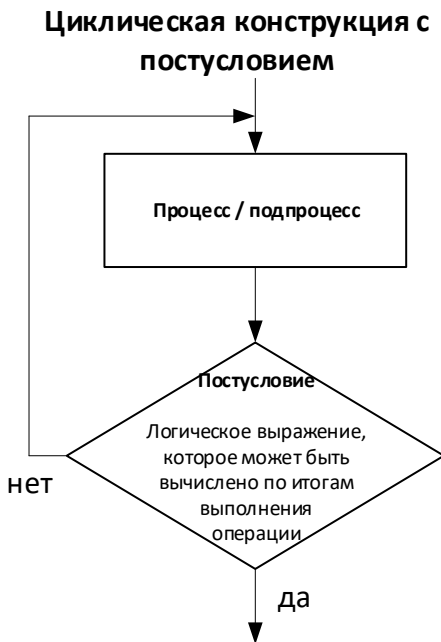


Рисунок 8.10 – Блоковая структура цикла с постусловием

В тех случаях, когда в программе применяется тот или иной метод, но при этом неизвестно содержимое этого метода, либо не требуется раскрытие содержимого метода, используется блок подпроцесса (Рисунок 8.11). В нём указывается лишь имя, характерное для метода, а также набор входных параметров, необходимых для нормальной работы метода (если таковые имеют место). При необходимости метод, указанный на основной блок-схеме как подпроцесс, может быть описан дополнительной блок-схемой алгоритма, раскрывающей его содержимое.

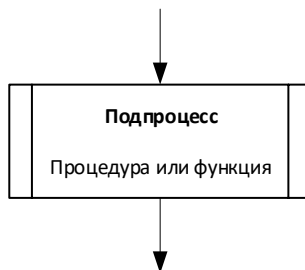


Рисунок 8.11 – Блок подпроцесса для обозначения используемых типизированных и не типизированных методов

Для случаев, когда операторы или логические выражения достаточно громоздки и заставляют растягивать блоки до довольно обширных размеров, предусмотрен особый блок комментариев. Как правило, грамотные составители блок-схем алгоритмов отказываются от изменения размеров блоков при невозможности уместить в них нужную текстовую конструкцию, а вводят ту или иную метку, которую определяют / раскрывают в комментарии, подключаемом к блоку так, как показано на Рисунке 8.12.

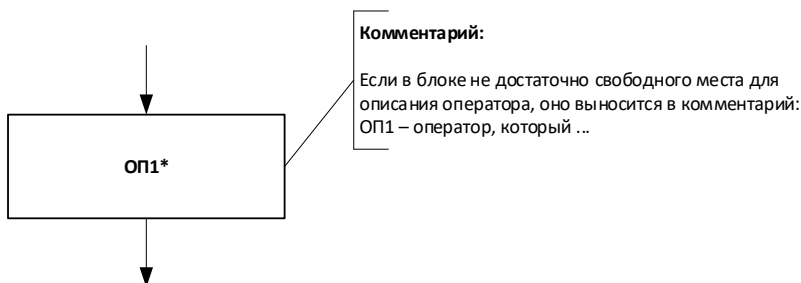


Рисунок 8.12 – Блок комментария

Когда же блок-схема получается достаточно длинной как по вертикали, так и по горизонтали, и размещение её на одной странице делает содержимое блоков мелким и совершенно не читаемым, грамотные составители применяют метод декомпозиции схемы и снабжают связанные переходы по ветвям ссылками. Ссылки позволяют разделить блок-схему на фрагменты, которые можно разместить либо в нескольких колонках на одной странице, либо же на разных страницах. Связываемые ссылками ветви блок-схемы именуются одинаково. Метки внутри ссылок могут быть как литерами, так и нумераторами (Рисунок 8.13).

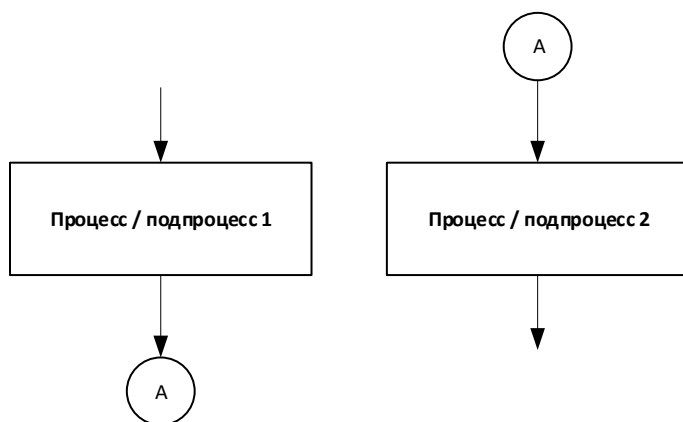


Рисунок 8.13 – Блок ссылки, используемый для переноса фрагмента схемы на другую страницу или в соседнюю колонку документа (метод декомпозиции схемы)

9 Примеры выполнения содержательной части задания

9.1. Задан фрагмент кода:

```
.....
//Пробуем открыть соединение с базой данных
try
{
    //Открываем соединение с базой данных
    c.Open();
}
//Если не получилось, то сбросить флаг готовности
//и уведомить пользователя
catch
{
    MessageBox.Show("Загрузка базы данных
прекращена.", "Не удалось открыть соединение с базой данных");
    ready = false;
}

if (ready)
{
    //Заполняем таблицы (элементы DataTable)
    CreateInnerTables(ref ready);
}

if (ready)
{
    //Проверяем, есть ли требуемые для расчёта атрибуты, а
    //если нет, то создаём их
    CheckFieldsDB(Path, ref ready);
}
.....
```

Изобразите фрагмент блок-схемы к нему.

Выполнение:

А. Фрагмент кода вырван из контекста, о чём свидетельствуют многоточия, указанные в начале и в конце кода. В связи с этим окончательные фигуры с надписями «Начало» и «Конец» изображать не требуется.

Б. Присутствует оператор отлова ошибок *try ... catch ... finally* с отсутствующим блоком *finally*. Изображать его следует блоком «Решение» с разветвлением вычислительного процесса.

В. Блок *catch*, реализующий разветвление в случае возникновения ошибки, содержит вывод на экран побуждающего сообщения для пользователя. Необходим блок «Документ».

Г. Завершают фрагмент два следующих подряд блока разветвления вычислительного процесса *if ... else ...*. С отсутствующим блоком *else*. Необходимы следующие подряд блоки «Решение».

Д. Блоки *if* содержат вызовы методов с именами «*CreateInnerTables*» и «*CheckFieldsDB*». Эти методы не являются стандартными методами языка программирования. Вероятно, они имеют определённое описание, которое не отражено в представленном фрагменте исходного кода, потому они оформляются блоками «Подпроцесс».

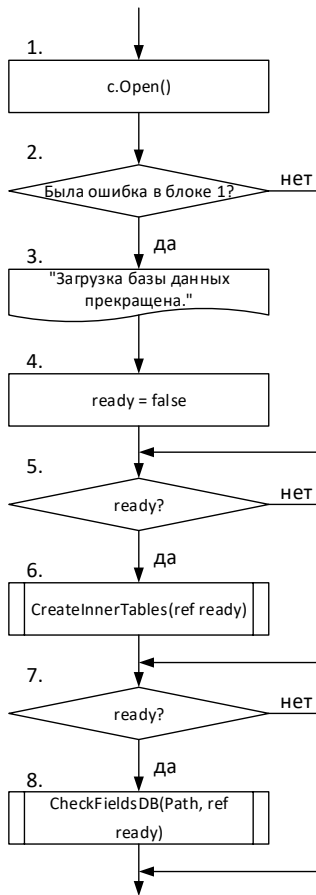


Рисунок 9.1 – Блок-схема алгоритма, соответствующая фрагменту программы

9.2. Задан код метода:

```
private void btnSave_Click(object sender, EventArgs e)
{
    mdlData.colDuty[cmbDutyList.SelectedIndex].Duty=
    txtDutyFromList.Text;
    mdlData.colDuty[cmbDutyList.SelectedIndex].Short=
    txtShort.Text;
    FillDutyList();
}
```

Изобразите блок-схему к нему.

Выполнение:

А. Приведён метод обработки события, связанного с нажатием кнопки мыши («*Click*») по интерфейсной кнопке с именем «*btnSave*». Оконечные фигуры «Начало/конец» необходимы.

Б. Обработка события состоит из двух простых операций и вызова метода, потому необходимы два блока «Процесс» и один блок «Подпроцесс».

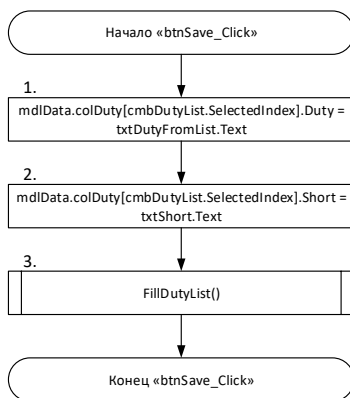


Рисунок 9.2 – Блок-схема алгоритма метода «*btnSave_Click*»

9.3. Задан фрагмент кода метода:

```
private void FillDutyList()
{
    int NumFix = cmbDutyList.SelectedIndex;

    //Очищаем список
    cmbDutyList.Items.Clear();

    //Заполняем комбо-список должностями
    for (int i = 0; i <= mdlData.colDuty.Count - 1; i++)
    {
        cmbDutyList.Items.Add(mdlData.colDuty[i].Code + ". " +
                               mdlData.colDuty[i].Duty);
    }
    .....
    Изобразите блок-схему к нему.
```

Выполнение:

А. Приведён фрагмент метода с именем «*FillDutyList*». Оконечная фигура «Начало/конец» необходима только для обозначения начала метода.

Б. Фрагмент метода содержит циклическую конструкцию. Цикл по фиксированному диапазону значений, потому для его отображения используется «Шестиугольник». Обратите внимание, что для иных циклических конструкций с пред- и постусловиями «Шестиугольник» неприменим, для их отображения должны быть использованы блоки «Решение».

В. В методе присутствуют локальные переменные, которые объявляются в этом методе с присвоением им вполне определённых начальных значений. Переменная *NumFix* объявляется в начале метода, параметр цикла *i* объявляется в перечне параметров цикла *for*.

Г. В теле цикла заключён обычный процесс, состоящий из одного оператора.

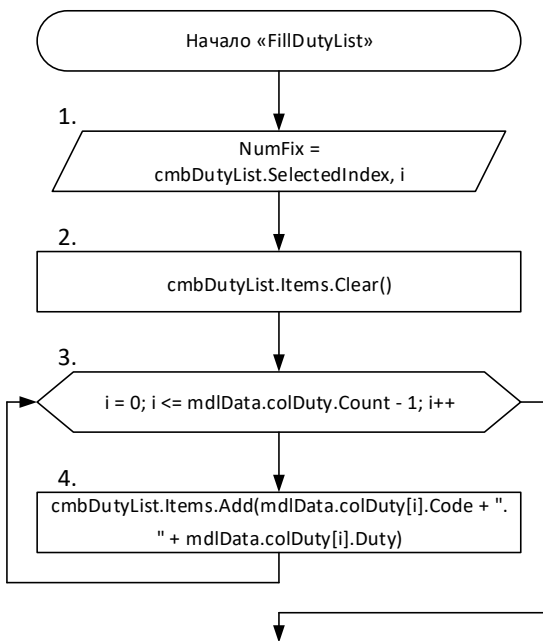


Рисунок 9.3 – Блок-схема алгоритма фрагмента метода
«*FillDutyList*»

Список литературы:

1. Подбельский, В.В. Язык C#. Базовый курс / В.В. Подбельский. – М.: Издательский дом «Инфра-М», 2011. – 382 с.
2. Шарп, Д. *Microsoft Visual C#*. Подробное руководство. 8-е издание / Д. Шарп / СПб.: «Питер», 2017. – 848 с.
3. *Microsoft*. [Электронный ресурс] : Создание оглавления - Word. URL: <https://support.office.com/ru-ru/article/Создание-оглавления-882e8564-0edb-435e-84b5-1d8552ccf0c0> (дата обращения: 01.10.2018).
4. Википедия. Свободная энциклопедия [Электронный ресурс] : Алгоритм. URL: <https://ru.wikipedia.org/wiki/Алгоритм> (дата обращения: 01.10.2018).
5. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. – М.: Государственный стандарт союза ССР, 1992. – 21 с.
6. GitHub. [Электронный ресурс] : Стиль библиографии для Word по ГОСТ Р ИСО 7.0.5 2008. URL: <https://github.com/irandom/docs> (дата обращения: 02.10.2018).

Приложение 1. Варианты индивидуального задания:

Вариант 1.

```
.....  
    //Цикл по всем таблицам базы данных согласно модели  
    for (int i = 0; i <= masTabNames.Length - 1; i++)  
    {  
        //Получение имени таблицы  
        tabName = masTabNames[i][0][0];  
        //Получение ключевого поля  
        tabKey = masTabNames[i][1][0];  
        //Попытка обращения с запросом на полную выборку  
        TryToSelectAll(tabName);  
    }  
.....
```

Вариант 2.

```
.....  
    if (MessageBox.Show("Создать таблицу " + tabName + "?",  
        "Попытка запустить систему с открытой БД",  
        MessageBoxButtons.YesNo) == DialogResult.Yes)  
    {  
        //Предпринимаем попытку создать таблицу базы данных  
        //Если дан положительный ответ, то  
        TryToCreateTable(tabName, tabKey);  
    }  
    //Если дан отрицательный ответ, то прекратить открытие базы данных  
    //- вылететь с ошибкой  
    else  
    {  
        ready = false;  
    }  
.....
```

Вариант 3.

```
    /// <summary>  
    /// Создание таблиц базы данных в оперативной памяти  
    /// </summary>  
    private void CreateInnerTables(ref bool ready)  
    {  
        DataTable Tab;  
        for (int i = 0; i <= masTabNames.Length - 1; i++)  
        {  
            Tab = getDataTableByName(masTabNames[i][0][0]);  
            InnerTableFill(masTabNames[i][0][0], ref Tab, masTabNames[i][2]);  
            setDataTableByName(masTabNames[i][0][0], Tab);  
        }  
    }  
}
```

Вариант 4.

```
.....  
    if (Path != mdlData.DataBasePath)  
    {  
        try  
        {  
            File.Copy(mdlData.DataBasePath, Path);  
        }  
        catch { }  
    }  
.....
```

Вариант 5.

```
.....  
    if (InitialBack)  
    {  
        if (Spaces)  
        {  
            str = Surname + " " + Name + " " + Patronymic;  
        }  
        else  
        {  
            str = Surname + " " + Name + Patronymic;  
        }  
    }  
    else  
    {  
        if (Spaces)  
        {  
            str = Name + " " + Patronymic + " " + Surname;  
        }  
        else  
        {  
            str = Name + Patronymic + " " + Surname;  
        }  
    }  
.....
```

Вариант 6.

```
public static string ExcelCellTranslator(int i, int j)
{
    string cell = "";
    int x;
    int lose;

    x = j;

    if (x < 16384)
    {
        lose = (x - 1) / 676;
        if (lose > 0)
        {
            cell += Alphabet(lose);
            x = x - (676 * lose);
        }
        lose = (x - 1) / 26;
        if (lose > 0)
        {
            cell += Alphabet(lose);
            x = x - (26 * lose);
        }
        cell += Alphabet(x);
    }
    else
    {
        cell += "XFD";
    }
    cell += i.ToString();
    return cell;
}
```

Вариант 7.

```
.....
    flgExist = false;
    //Перебираем дополнительную работу
    for (int k = 0; k <= colDopWork.Count - 1; k++)
    {
        if (colDopWork[k].Lecturer.Equals(DW.Lecturer) &
            colDopWork[k].Semestr.Equals(DW.Semestr))
        {
            //Если нашлось совпадение
            flgExist = true;
        }
    }
    .....
}
```

Вариант 8.

```
.....
bool flgComplexDiplomaInHoured = false;
string DiplomaMask = "";
string SubMask = "";
string Lect = "";
int Mask = 0;
int Iter = 0;

clsDistribution D;
//Очищаем коллекцию штатной нагрузки с учётом почасовой
OutColl.Clear();
//Заполняем коллекцию новыми объектами
for (int i = 0; i <= InColl.Count - 1; i++)
{
    D = new clsDistribution();
    OutColl.Add(D);
}
.....
```

Вариант 9.

```
public static string DateToSQL(DateTime D)
{
    string str = "";

    if (D.Month < 10)
    {
        str = "#0" + D.Month + "/";
    }
    else
    {
        str = "#" + D.Month + "/";
    }
    if (D.Day < 10)
    {
        str += "0" + D.Day + "/";
    }
    else
    {
        str += D.Day + "/";
    }
    str += D.Year + "#";
    return str;
}
```

Вариант 10.

```
.....
if (ready)
{
    // Пытаемся выполнить очистку содержимого
    try
    {
        for (int i = 0; i <= masTabNames.Length - 1; i++)
        {
            InnerDelete(masTabNames[i][0][0], masTabNames[i][1][0]);
        }
    }
    //Обрабатываем исключение в случае возникновения ошибки и
    //фиксируем ошибку снятием флага готовности и закрываем соединение
    catch (OleDbException e)
    {
        MessageBox.Show(e.Message, "Ошибка");
        ready = false;
    }
    c.Close();
}
.....
```

Вариант 11.

```
.....
if (coll.Visiting > 0)
{
    if (!coll.flgDistrib)
    {
        subjInfo += "посещ.-" + coll.Visiting.ToString() + "; ";
    }
    else
    {
        subjInfo += "посещ.-" + coll.Weight.ToString() + "н/чел.; ";
    }
}
.....
```

Вариант 12.

```
.....
flgExist = false;
for (int m = 0; m <= colSchedule.Count - 1; m++)
{
    if (colSchedule[m].Lecturer.FIO.Equals(S.Lecturer.FIO) &
        colSchedule[m].Time.Time.Equals(S.Time.Time) &
        colSchedule[m].Week.NumberWeek.Equals(S.Week.NumberWeek)
        & colSchedule[m].WeekDay.WeekDay.Equals(S.WeekDay.WeekDay))
    {
        flgExist = true;
    }
}
.....
```


Вариант 13.

```
public static bool NonZeroForDispatchOR(clsDistribution coll)
{
    bool flg = coll.Lecture > 0 ||
               coll.Practice > 0 ||
               coll.LabWork > 0 ||
               coll.KursProject > 0 ||
               coll.TutorialPractice > 0 ||
               coll.PreDiplomaPractice > 0 ||
               coll.ProducingPractice > 0;

    return flg;
}
```

Вариант 14.

```
public static int CountStringOccurrences(string Text, string Pattern)
{
    int count = 0;
    int i = 0;

    while ((i = Text.IndexOf(Pattern, i)) != -1)
    {
        i += Pattern.Length;
        count++;
    }

    return count;
}
```

Вариант 15.

```
/// <summary>
/// Событие, связанное с загрузкой главной формы
/// </summary>
/// <param name="sender">Объект, содержащий сведения об источнике вызова
/// обработчика этого события</param>
/// <param name="e">Перечень параметров среды, связанных с событием
/// загрузки главной формы</param>
private void frmMain_Load(object sender, EventArgs e)
{
    InitMain();
}
```

Вариант 16.

```
.....
//Неделя
GetCode = 0;
Detected = false;
for (int i = 0; i <= mdlData.colWeek.Count - 1; i++)
{
    CurrentString = Tab.Rows[id]["Неделя"].ToString();
    if (mdlData.colWeek[i].NumberWeek == CurrentString)
    {
        GetCode = i;
        Detected = true;
    }
}
if (Detected)
{
    this.Week = mdlData.colWeek[GetCode];
}
else
{
    this.Week = null;
    MessageBox.Show("Не удалось определить неделю у элемента с кодом " +
        this.Code, "Оповещение");
}
.....
```

Вариант 17.

```
void frmLecturerID_Resize(object sender, EventArgs e)
{
    if (this.Width >= fWidth & this.Height >= fHeigth)
    {
        dgNagruzka.Width = this.Width - 40;

        btnClose.Top = this.Height - 50 - btnClose.Height;
        btnClose.Left = this.Width - 30 - btnClose.Width;
        cmbForm.Left = dgNagruzka.Left;
        cmbForm.Top = btnClose.Top;
        btnAction.Left = cmbForm.Left + cmbForm.Width + 10;
        btnAction.Top = cmbForm.Top;

        frParams.Top = cmbForm.Top - 10 - frParams.Height;
        frParams.Left = dgNagruzka.Left;

        dgNagruzka.Height = (frParams.Top - 10) - dgNagruzka.Top;
    }
    else
    {
        this.Width = fWidth;
        this.Height = fHeigth;
    }
}
```

Вариант 18.

```
.....
//Задаём количество столбцов
//оно остаётся неизменным
//в количестве 6 штук
for (int i = 0; i <= 5; i++)
{
    dgNagruzka.Columns.Add("", "");
}

countRow = countTableRowsDiploma(coll, cmbLecturerList.SelectedIndex);

for (int i = 0; i <= countRow - 2; i++)
{
    dgNagruzka.Rows.Add();
}

DataGridFillerDiploma(coll);
.....
```

Вариант 19.

```
.....
//-----ПИШЕМ ПОЛНОЕ НАЗВАНИЕ КАФЕДРЫ
if (!chkAfter2015.Checked)
{
    if (!flgBoss)
    {
        spacesBefore(ObjMissing, ObjDoc, 6);
    }
    else
    {
        spacesBefore(ObjMissing, ObjDoc, 8);
    }
}
else
{
    spacesBefore(ObjMissing, ObjDoc, 5);
}
.....
```

Вариант 20.

```
public static string getDoWString(DayOfWeek DoW)
{
    string str = "";

    switch (DoW)
    {
        case DayOfWeek.Monday:
            str = "в понедельник";
            break;
        case DayOfWeek.Tuesday:
            str = "во вторник";
            break;
        case DayOfWeek.Wednesday:
            str = "в среду";
            break;
        case DayOfWeek.Thursday:
            str = "в четверг";
            break;
        case DayOfWeek.Friday:
            str = "в пятницу";
            break;
        case DayOfWeek.Saturday:
            str = "в субботу";
            break;
        case DayOfWeek.Sunday:
            str = "в воскресенье";
            break;
    }

    return str;
}
```

Вариант 21.

```
private void intoWord()
{
    //Задаём переменную для отсутствующего параметра
    object ObjMissing = Missing.Value;

    try
    {
        //Создаём новое Word приложение
        Word._Application ObjWord = new Word.Application();

        wordCore(ObjMissing, ObjWord);

        ObjWord.Quit();
    }
    catch
    {
        MessageBox.Show("Возможно на этом компьютере присутствует проблема
        совместимости с MS Word." +
        " Попробуйте установить версию 2007 и выше.");
    }
}
```

Вариант 22.

```
.....
//Упорядочивание по убыванию возможности замены преподавателями
for (int i = 0; i <= mdlData.colLectLoad.Count - 2; i++)
{
    for (int j = i + 1; j <= mdlData.colLectLoad.Count - 1; j++)
    {
        //Если впереди (справа от текущего i) элемент с большими
        возможностями, его необходимо поднять по списку (повысить
        приоритет)
        if (mdlData.colLectLoad[i].Load < mdlData.colLectLoad[j].Load)
        {
            //Промежуточный объект-переменная
            clsLecturer_Load LLtmp = new clsLecturer_Load();
            LLtmp = mdlData.colLectLoad[i];
            mdlData.colLectLoad[i] = mdlData.colLectLoad[j];
            mdlData.colLectLoad[j] = LLtmp;
            LLtmp = null;
        }
    }
}
for (int i = 0; i <= mdlData.colLectLoad.Count - 1; i++)
{
    lstSwap.Items.Add("Частично может заменить: " +
    mdlData.colLectLoad[i].Lecturer.FIO + " - " +
    Convert.ToInt32((Convert.ToDouble(mdlData.colLectLoad[i].Load) /
    Convert.ToDouble(unitCountIll)) * 100) + "%");
}
.....
```

Вариант 23.

```
private void textDepartName(object ObjMissing, Word._Document ObjDoc)
{
    object ObjRange;
    //Задаём закладку конца документа
    object EndOfDoc = "\\endofdoc";
    Word.Paragraph ObjParagraph;

    ObjRange = ObjDoc.Bookmarks.get_Item(ref EndOfDoc).Range;
    ObjParagraph = ObjDoc.Content.Paragraphs.Add(ref ObjRange);
    ObjParagraph.Range.Text = "Кафедра \\\"Управление и защита информации\\\"";
    //Отступа слева нет
    ObjParagraph.Format.LeftIndent = 0;
    //Жирный шрифт
    ObjParagraph.Range.Font.Bold = 1;
    //Все обычные, не заглавные
    ObjParagraph.Range.Font.AllCaps = 0;
    //Выравнивание по центру
    ObjParagraph.Alignment =
    Word.WdParagraphAlignment.wdAlignParagraphCenter;
    //Добавляем ещё один абзац текста
    ObjParagraph.Range.InsertParagraphAfter();
}
```

Вариант 24.

```
.....
    for (int i = 0; i <= dgNagruzka.RowCount - 1; i++)
    {
        for (int j = 0; j <= dgNagruzka.ColumnCount - 1; j++)
        {
            if (!(dgNagruzka[j, i].Value == null))
            {
                ObjTable.Cell(i + 1, j + 1).Range.Text = dgNagruzka[j, i].Value.ToString();
            }
        }
    }
}
.....
```

Вариант 25.

```
private void allIntoWord()
{
    //Задаём переменную для отсутствующего параметра
    object ObjMissing = Missing.Value;
    try
    {
        //Создаём новое Word приложение
        Word._Application ObjWord = new Word.Application();

        for (int k = 0; k <= cmbLecturerList.Items.Count - 1; k++)
        {
            cmbLecturerList.SelectedIndex = k;

            if (mdlData.colLecturer[k].Rate > 0)
            {
                wordCore(ObjMissing, ObjWord);
            }
        }
        ObjWord.Quit();
    }
    catch
    {
        MessageBox.Show("Возможно на этом компьютере присутствует проблема
        совместимости с MS Word." +
        " Попробуйте установить версию 2007 и выше.");
    }
}
```

Вариант 26.

```
/// <summary>
/// Настройка начального состояния элементов управления формы
/// </summary>
private void InitMain()
{
    //Без подгруженной базы данных
    //Ничего кроме закрытия программы и открытия
    //базы данных сделать нельзя
    btnClose.Enabled = true;
    btnLoad.Enabled = true;
    chkOldDB.Enabled = true;

    btnSave.Enabled = false;
    btnReset.Enabled = false;
    frEdit.Enabled = false;
    frEditComplex.Enabled = false;
    frFeatures.Enabled = false;
    frOutput.Enabled = false;
}
```

Приложение 2. Методика составления автособираемого библиографического списка

Как было отмечено во Введении данного издания, создание автособираемого библиографического списка не является обязательным элементом Задания 1 Учебной практики, потому оно вынесено в Приложение.

Дополнительной причиной, по которой рассматриваемая методика является факультативной для обучающихся младших курсов, является тот факт, что отдельные материалы, собранные в этом Приложении, требуют загрузки файлов из частного Интернет-источника, который со временем может стать недоступным. Не исключено, что будут созданы иные, аналогичные источники, поскольку в них представлена, действительно, нужная и важная информация с точки зрения составления отчётной документации.

Начнём рассмотрение издалека: в любом из известных браузеров перейдём на сайт <https://github.com/irandom/docs> [6]. Это частный раздел ресурса, именуемого *GitHub*. *GitHub* – крупнейший веб-сервис для хранения на нём программных проектов и их совместной разработки. Веб-сервис основан на системе контроля версий *Git*, он бесплатен для проектов с открытым исходным кодом и предоставляет им все возможные привилегии. Для частных же, закрытых проектов предлагаются различные платные тарифные планы. Рассматриваемый проект с ГОСТами оформления библиографических списков на верхнем уровне состоит из двух элементов: директории «*gost-r-7.0.5-2008*» и текстового файла с описанием проекта (Рисунок П2.1).

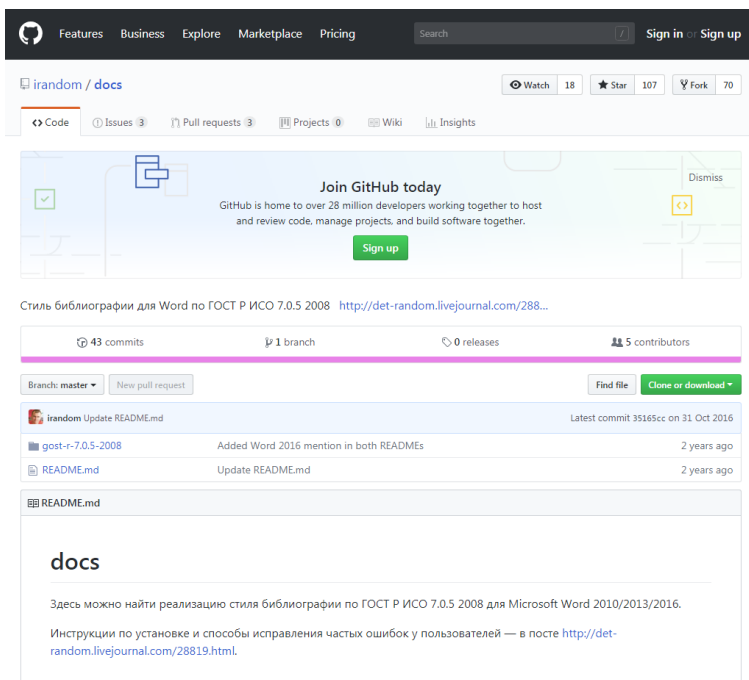


Рисунок П2.1 – Веб-страница <https://github.com/irandom/docs>, содержащая шаблоны ГОСТов для оформления библиографических списков в *Microsoft Office Word*

В правом верхнем углу таблицы, содержащей перечень файлов проекта, расположена зелёная кнопка клонирования или загрузки проекта на жёсткий диск (*Clone or download*). Нажатие на кнопку разворачивает диалоговое окно, в нижней правой части которого размещена ещё одна кнопка «Скачать ZIP» (*Download ZIP*). Это и есть необходимая для скачивания файлов ссылка (Рисунок П2.2). Вложенная конструкция оберегает посетителей сайта от загрузки проекта случайным щелчком мышью по зелёной кнопке.

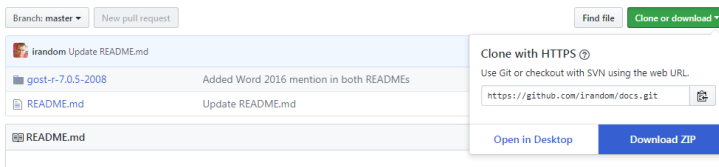


Рисунок П2.2 – Переход к скачиванию архива с шаблонами ГОСТов для оформления библиографических списков в *Microsoft Office Word*

На жёсткий диск скачивается архив *docs-master.zip*, содержащий внутри ту же структуру, которая представлена на сайте *GitHub*. Раскроем директорию *gost-r-7.0.5-2008*, где хранятся три файла: два стилевых (необходимых для работы) и один текстовый (вспомогательный) с пользовательской инструкцией по их установке (Рисунок П2.3). Стилевые файлы составлены в формате *XSL (eXtensible Stylesheet Language)*, представляющем собой семейство разметки с детальным описанием правил визуализации *XML*-документов. Написание управления документами в таком формате не случайно для *Microsoft Office Word*. Отметим, что если переименовать документ формата **.docx* в **.zip*, распаковать этот архив и «пробежаться» по его структуре, то можно увидеть, что он состоит, преимущественно, из *XML*-документов.

c:\Users\Flash_A\Downloads\docs-master.zip\docs-master\gost-r-7.0.5-2008*.*				
Имя	Тип	Размер	Тип файла	Дата изменения
..				
GOST-R-7.0.5-2008.xsl		158,9 Кб		31.10.2016 03:59:16
GOST-R-7.0.5-2008-lexicographically.xsl		159,5 Кб		31.10.2016 03:59:16
README.md		4,2 Кб		31.10.2016 03:59:16

Рисунок П2.3 – Содержимое загруженного на жёсткий диск архива *docs-master.zip*

Рассмотрим фрагмент текстового редактора *Microsoft Office Word*, к которому и запланировано подключение загруженных

файлов. Откроем *Microsoft Office Word* и развернём панель инструментов, содержащуюся в пункте меню «Ссылки». Приблизительно посередине панели инструментов размещается блок/раздел «Ссылки и списки литературы», содержащий один единственный комбинированный список напротив надписи «Стиль». Развернём его. На Рисунке П2.4 показано, что в стандартной комплектации для пользователя представлено 12 наименований различных стилей оформления библиографических списков.

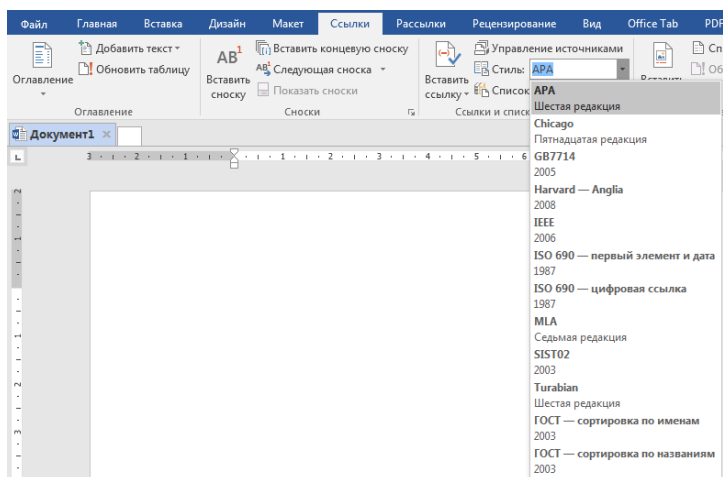
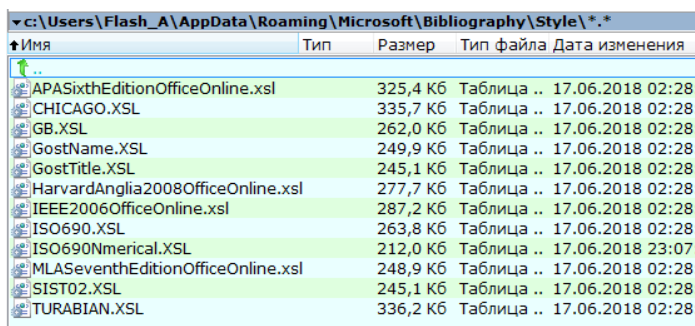


Рисунок П2.4 – Стандартный перечень стилей оформления библиографических списков *Microsoft Office Word*

Те же 12 стилей оформления библиографических списков содержатся в специальной (и даже скрытой) директории, соответственно, со стилями оформления библиографии в составе текстового редактора *Microsoft Office Word* (Рисунок П2.5). Они вложены, что характерно, логично и последовательно один в другой: «*Bibliography* > *Style*». Воспользуемся подсказкой, составленной автором загруженных

на жёсткий диск стилей. Он резонно заметил, что в зависимости от версии *Microsoft Office* место расположения стилей существенно меняется [6]:

- для *Microsoft Office Word 2010* – это путь [System Volume]:\Program Files\Microsoft Office\Office 14\Bibliography\Style (открытый путь);
- для *Microsoft Office Word 2013 (2016)* – это путь [System Volume]:\Users\[User Name]\AppData\Roaming\Microsoft\Bibliography\Style (AppData – скрытое звено пути).



Имя	Тип	Размер	Тип файла	Дата изменения
APASixthEditionOfficeOnline.xsl		325,4 Кб	Таблица ..	17.06.2018 02:28
CHICAGO.XSL		335,7 Кб	Таблица ..	17.06.2018 02:28
GB.XSL		262,0 Кб	Таблица ..	17.06.2018 02:28
GostName.XSL		249,9 Кб	Таблица ..	17.06.2018 02:28
GostTitle.XSL		245,1 Кб	Таблица ..	17.06.2018 02:28
HarvardAnglia2008OfficeOnline.xsl		277,7 Кб	Таблица ..	17.06.2018 02:28
IEEE2006OfficeOnline.xsl		287,2 Кб	Таблица ..	17.06.2018 02:28
ISO690.XSL		263,8 Кб	Таблица ..	17.06.2018 02:28
ISO690Nmerical.XSL		212,0 Кб	Таблица ..	17.06.2018 23:07
MLASeventhEditionOfficeOnline.xsl		248,9 Кб	Таблица ..	17.06.2018 02:28
SIST02.XSL		245,1 Кб	Таблица ..	17.06.2018 02:28
TURABIAN.XSL		336,2 Кб	Таблица ..	17.06.2018 02:28

Рисунок П2.5 – Исходный перечень файлов стилей оформления библиографических списков *Microsoft Office Word*, размещённых в специальной директории *Microsoft Office 2016*

Скопируем два новых стиля оформления библиографии в директорию, после чего количество файлов в ней увеличится до 14 штук (Рисунок П2.6). Если текстовый редактор *Microsoft Office Word* до момента копирования файлов в директорию был открыт, то его необходимо перезапустить для того, чтобы выполненные изменения вступили в силу.

c:\Users\Fish_A\AppData\Roaming\Microsoft\Bibliography\Style*.*				
Имя	Тип	Размер	Тип файла	Дата изменения
..				
APASixthEditionOfficeOnline.xsl		325,4 Кб	Таблица ..	17.06.2018 02:28
CHICAGO.XSL		335,7 Кб	Таблица ..	17.06.2018 02:28
GB.XSL		262,0 Кб	Таблица ..	17.06.2018 02:28
GOST-R-7.0.5-2008.xsl		158,9 Кб	Таблица ..	31.10.2016 03:59
GOST-R-7.0.5-2008-lexicographically.xsl		159,5 Кб	Таблица ..	31.10.2016 03:59
GostName.XSL		249,9 Кб	Таблица ..	17.06.2018 02:28
GostTitle.XSL		245,1 Кб	Таблица ..	17.06.2018 02:28
HarvardAnglia2008OfficeOnline.xsl		277,7 Кб	Таблица ..	17.06.2018 02:28
IEEE2006OfficeOnline.xsl		287,2 Кб	Таблица ..	17.06.2018 02:28
ISO690.XSL		263,8 Кб	Таблица ..	17.06.2018 02:28
ISO690Nmerical.XSL		212,0 Кб	Таблица ..	17.06.2018 23:07
MLASeventhEditionOfficeOnline.xsl		248,9 Кб	Таблица ..	17.06.2018 02:28
SIST02.XSL		245,1 Кб	Таблица ..	17.06.2018 02:28
TURABIAN.XSL		336,2 Кб	Таблица ..	17.06.2018 02:28

Рисунок П2.6 – Дополненный перечень файлов стилей оформления библиографических списков *Microsoft Office Word*

После перезагрузки текстового редактора можно наблюдать в меню «Ссылки» автоматическое расширение перечня стилей оформления библиографических списков. Выберем в нём один из новых стилей отвечающий, например, последним требованиям составления библиографических списков в научных публикациях «ГОСТ Р 7.0.5.-2008 (сортировка по порядку включения)» (Рисунок П2.7).

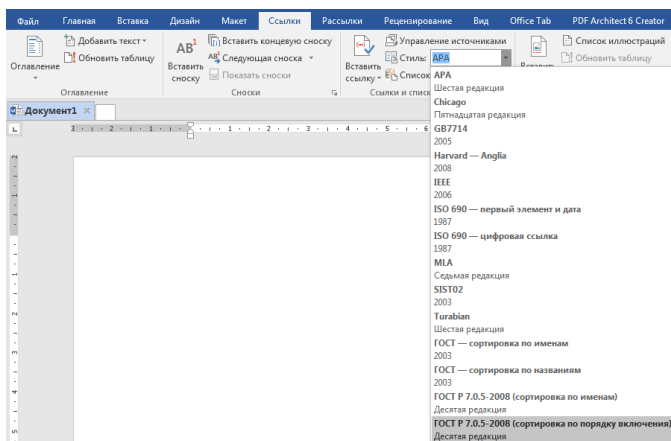


Рисунок П2.7 – Расширенный перечень стилей оформления библиографических списков в меню «Ссылки» *Microsoft Office Word*

Рассмотрим далее решение задачи составления автособираемого библиографического списка на примере обновления библиографического списка в документе *Microsoft Office Word*. Допустим, список был составлен вручную в текстовом редакторе предыдущего поколения (*Microsoft Office Word* до версии 2007 года).

Если файл был создан в одной из версий *Microsoft Office Word* предыдущего поколения, то он гарантированно хранится на жёстком диске в формате *.doc и обладает при открытии в новых версиях, так называемой, ограниченной функциональностью. Для расширения функциональности документа необходимо перейти в меню «Файл» и в разделе «Сведения» нажать на кнопку «Преобразовать» (Рисунок П2.8), после чего документ необходимо сохранить под другим или под тем же именем, но уже в формате *.docx.

Отметим, что в формате документа *.doc операции, связанные с составлением библиографического списка, работать всё равно будут, но не в полном объёме. Этот факт не трудно проверить на практике – вставленные согласно изложенной методике элементы будут статичны (далее наряду с этим термином будет использован сленг – «мёртвые» ссылки), в то время как в документе, отвечающем последним требованиям форматирования, они будут динамичны (наряду с термином будет использован сленг – «живые» ссылки).

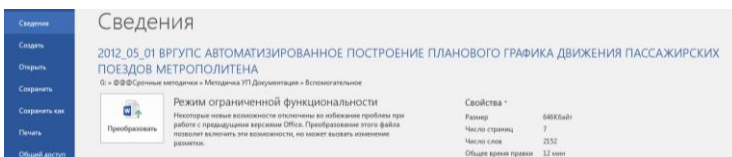


Рисунок П2.8 – Для полноценной работы с элементами автособираемого библиографического списка старые файлы *.doc должны быть преобразованы в новые *.docx

Смена формата файла потребует согласия пользователя и подтверждения им представленных на Рисунке П2.9 положений. Действительно, в преобразованном файле могут произойти незначительные изменения в форматировании текста, например «схлопывание пробелов» между латиницей и кириллицей, потому после преобразования документов не в учебных целях их рекомендуется повторно вычитать.

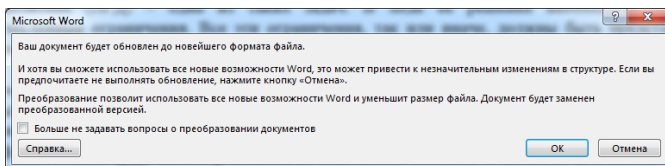


Рисунок П2.9 – Предупреждение о смене формата документа
Microsoft Office Word

Итак, в качестве образцовой работы по составлению автособираемого библиографического списка рассматривается научная публикация, фрагменты которой представлены далее на Рисунках П2.10 – П2.12. Эти фрагменты демонстрируют ссылки, выполненные в тексте документа – заключённые в квадратные скобки численные значения (Рисунки П2.10 и П2.11), направляющие к прочтению библиографических источников, собранных в конце статьи (Рисунок П2.12).

В статье [1] авторами была сформулирована постановка задачи автоматизированного построения ПГД в общем виде и записана следующим образом: разработать алгоритмы, согласно которым, в результате конечного числа ответов пользователя на общие вопросы (с вариантами ответов «Да/Нет») с промежуточным вводом исходных данных будет построен ПГД при учёте специфических параметров и ограничений, в первую очередь, определяемых графиком оборота подвижного состава (ГО), который регулирует проведение осмотров и ремонтов подвижного состава, и графиком работы локомотивных бригад. Построенный ПГД должен отвечать поставленным целям управления, быть рациональным с точки зрения выбранных критериев и устойчивым к возмущающим факторам.

Алгоритмы автоматизированного построения ПГД представляют собой сценарии управления объектами линии метрополитена. Эти алгоритмы реализуют рациональные управляющие воздействия для каждого из процессов ПГД.

Управляющими воздействиями являются императивы и логико-трансформационные правила (ЛТП) построения ПГД [2]. Определение объектов, к которым они применяются, и построение логики их выполнения проводится на базе предварительного расчёта. В ходе расчёта используются введенные пользователем данные, проводится оценка графика по выбранным критериям, учитываются действующие ограничения.

Рисунок П2.10 – Первый фрагмент текста, подлежащего форматированию ссылок на литературу

на линии, имеют свои характерные особенности [1].

Авторами предложено оценивать достижение поставленных целей управления при помощи условий реализации. Под условиями реализации понимается апостериорная информация, получаемая по итогам построения переходного/установившегося процесса путём проверки графика после завершения рассматриваемого процесса. Условия реализации позволяют определить, удалось ли построить процесс при заданных начальных условиях. В этом случае, термин «условия реализации» употребляется применительно к построению отдельных процессов ПГД, а термин «условия успешной реализации» – применительно к ПГД, составленному на весь день.

Опыт эксплуатации линий Московского метрополитена показал, что использование предельных, с точки зрения безопасности движения, значений парности движения может привести к частому возникновению сбоев в движении поездов. Это, как правило, связано с воздействием такого возмущающего фактора, как задержка поезда пассажирами. В связи с этим, принятие решений о реализации предельных значений парности движения, должно подкрепляться предварительным проведением имитационных экспериментов [3].

Ограничения, накладываемые на ПГД, обусловлены общими и технологическими требованиями обеспечения безопасности движения поездов, а также связями между объектами линии. К ним относятся:

- порядок заполнения точек ночной расстановки составов на линии;
- возможность проведения регулировочных действий на станциях с путевым развитием;
- частота ввода и снятия составов на промежуточных станциях в соответствии с правилами обслуживания пассажиров;
- правила функционирования станций с путевым развитием;
- время окончания движения;
- время отправления последних пассажирских поездов с начальных станций путей;
- организация движения последних пассажирских поездов [4].

Рисунок П2.11 – Второй фрагмент текста, подлежащий форматированию ссылок на литературу

Литература:

1. Сидоренко В.Г., Сафронов А.И. Построение планового графика движения для метрополитена // Мир транспорта. 2011, № 3. - С. 98-105.
2. Сидоренко В.Г. Автоматизация построения планового графика движения поездов метрополитена // Автоматизация и современные технологии, 2003, №2, С. 6–10.
3. Баранов Л.А., Сидоренко В.Г. Тренажер поездных диспетчеров линий Московского метрополитена // Железные дороги мира, 2002, №8. С. 64-69.
4. Пискунов А.С., Сидоренко В.Г. Процедуры организации ночной расстановки составов на линии метрополитена // ВЕСТНИК МИИТа // Научно-технический журнал. М.: МИИТ. 2008, вып. 18. - С. 3-7.

Рисунок П2.12 – Список литературы, составленный вручную в текстовом редакторе *Microsoft Office Word* предыдущего поколения (*.doc)

Сделаем отступ в несколько строк от существующего списка литературы и на той же панели инструментов, относящейся к меню «Ссылки» раскроем пункт «Список литературы». В нём выберем подходящий для работы шаблон. Вероятно, в списке он окажется первым, и называться будет так же, как и пункт панели инструментов «Список литературы» (Рисунок П2.13). После выбора пункта списка в области документа вслед за курсором будет вставлен блок автособираемого библиографического списка. Этот список окажется пустым, о чём возвестит системная надпись вида: «Текущий документ не содержит источников».

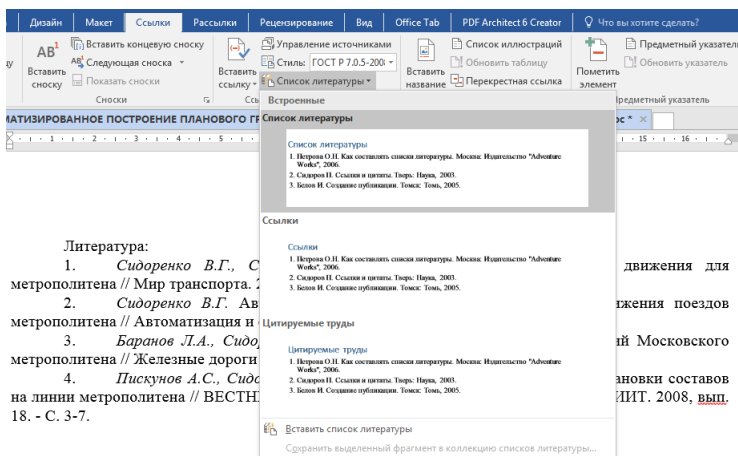
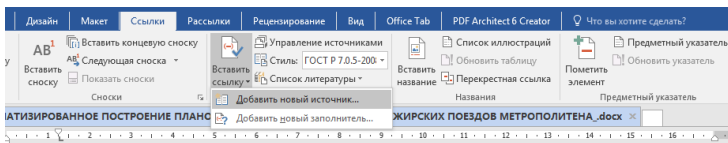


Рисунок П2.13 – Подготовка к вставке блока автособираемого библиографического списка в документ

Выделим первый пункт библиографического списка и скопируем его в буфер обмена. Далее снова обратим внимание на панель инструментов, содержащуюся в пункте меню «Ссылки». Раскроем там пункт «Вставить ссылку» и выберем «Добавить новый источник...» (Рисунок П2.14). Это наиболее быстрый способ перейти к редактированию нового источника (Рисунок П2.15). Однако, добавление источника сразу повлечёт за собой вставку ссылки на него в указанный курсором (или выделенный) фрагмент документа, потому в спокойной обстановке, не требующей спешки, рекомендуется поступать правильно и добавлять источники через специально предусмотренный для этих целей редактор, вызываемый через пункт «Управление источниками». Оттуда в раздел, представленный на Рисунке П2.15, можно перейти посредством нажатия на кнопку «Создать».



Литература:

1. *Сидоренко В.Г., Сафронов А.И.* Построение планового графика движения для метрополитена // Мир транспорта. 2011, № 3. - С. 98-105.

Рисунок П2.14 – Копирование первого элемента списка литературы и переход к созданию нового источника

Для создания нового источника необходимо в первую очередь определиться с его корректной идентификацией, с типом, к которому он относится. Это может быть книга, журнальная статья, электронный источник или нечто иное. Должно быть указано то, что согласно ГОСТу в состоянии однозначно определять вид упоминания этого источника в блоке автособираемого библиографического списка. И, естественно, важно не забыть вставить содержимое буфера обмена в одно из полей экранной формы «Создать источник» (Рисунок П2.15).

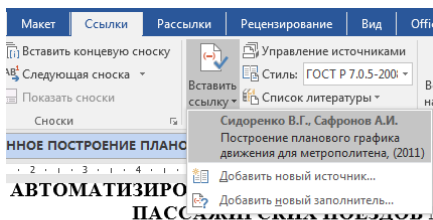
Рисунок П2.15 – Для удобства заполнения экранной формы содержимое буфера обмена вставляется в одно из полей, и далее разбирается по соответствующим полям

Предпочтительнее вставку из буфера обмена выполнить в поле с указанием автора / авторов, поскольку в скопированной строке информация об авторстве указывается в самом её начале. Далее необходимо постепенно вырезать оттуда с «хвоста» нужные элементы, вставляя их уже в соответствующие поля как показано на Рисунке П2.16. Избыточные разделительные символы из строки «Автор» нужно будет удалить по завершении распределения сведений. Обратите внимание, что если автор у источника один, то после последней точки в инициалах, необходимо ставить запятую. В ином случае пробел, разделяющий инициалы и фамилию, будет воспринят как разделитель. Так в блоке автособираемого библиографического списка можно увидеть, например, следующую конструкцию «Сидоренко, В.Г.». Однако, согласно требованиям некоторых из существующих ГОСТов, в документах требуется выполнять именно такое упоминание автора, ответственного за публикацию.

Рисунок П2.16 – Результат разбора первого элемента по полям

В результате создания литературного источника в структуре документа *Microsoft Office Word* на него появляются ссылки в элементах соответствующего меню «Ссылки» (Рисунок П2.17) и становится возможным его многократное использование в

тексте. Более того, вместе с этим автоматом учитывается порядок следования источников в тексте. **Важно:** за порядковыми номерами источников не нужно следить — это существенно экономит время.



современный мир информационных технологий при использовании человеческого труда на различных производствах далеко не всегда легко формализовать. Как из этого, может быть, лишь частным случаем решения. Процесс составления планового графика движения (ПГД) — одна из таких задач. В условиях жестких ограничений. Все эти ограничения, так же задачи.

В статье [1] авторами была сформулирована

Рисунок П2.17 – Подготовка к замене «мёртвой» ссылки на источник «живой» ссылкой на источник

«Живые» ссылки на литературные источники обладают собственными функциональными возможностями, такими как, например, обновление блока автособираемого библиографического списка, а также всех ссылок, имеющих в тексте. Это очень удобная функция, поскольку в таком случае не требуется прокрутка всего документа с целью обнаружения и обновления блока автособираемого библиографического списка. Достаточно найти хотя бы одну «живую» ссылку в тексте и выполнить обновление блока автособираемого библиографического списка через неё. Это накладывает определённый отпечаток — при выделении ссылка на литературу начинает выглядеть громоздко, но к этому можно привыкнуть и с этим можно смириться (Рисунок П2.18).

Процесс составления планового графика движения пассажирских поездов по линии метрополитена (ПГД) – одна из таких задач. В ходе её решения необходимо учитывать многочисленные ограничения. Все эти ограничения, так или иначе, должны быть представлены в постановке задачи.

В статье [11] авторами была сформулирована постановка задачи автоматизированного построения ПГД в общем виде и записана следующим образом: разработать алгоритмы, согласно

Рисунок П2.18 – Результат «оживления» ссылки на источник

По аналогии с первым пунктом в структуру документа *Microsoft Office Word* вводятся три оставшихся элемента библиографии (Рисунки П2.19 – П2.21).

Создать источник

Тип источника: Журнальная статья

Язык: По умолчанию

Поля списка литературы для ГОСТ Р 7.0.5-2008 (сортировка по порядку включения)

Автор: Сидоренко В.Г.

☐ Корпоративный Автор

Название: Автоматизация построения планового графика движения поездов метрополитена

Название журнала: Автоматизация и современные технологии

Год: 2003

Месяц:

Страницы: С. 6–10

Том:

Выпуск: №2

☐ Показать все поля списка литературы

Имя тега: Сид03

Пример. 12

OK Отмена

Рисунок П2.19 – Результат разбора второго элемента по полям

Создать источник

Тип источника: Журнальная статья

Язык: По умолчанию

Поля списка литературы для ГОСТ Р 7.0.5-2008 (сортировка по порядку включения)

Автор: Баранов Л.А., Сидоренко В.Г.

☐ Корпоративный Автор

Название: Тренажер поездных диспетчеров линий Московского метрополитена

Название журнала: Железные дороги мира

Год: 2002

Месяц:

Страницы: С. 64-69

Том:

Выпуск: №2

☐ Показать все поля списка литературы

Имя тега: Бар02

Пример. 12

OK Отмена

Рисунок П2.20 – Результат разбора третьего элемента по полям

Создать источник

Тип источника: Журнальная статья

Язык: По умолчанию

Поля списка литературы для ГОСТ Р 7.0.5-2008 (сортировка по порядку включения)

* Автор: Пискунов А.С., Сидоренко В.Г. [Изменить]

* Название: Процедуры организации ночной расстановки составов на линии метрополитена

* Название журнала: ВЕСТНИК МИИТа

Город: М.

* Год: 2008

* Месяц:

День:

* Страницы: С. 3-7.

Редактор: [Изменить]

Издательство: МИИТ

* ...

☒ Показать все поля списка литературы

Рекомендованное поле

Имя тега: Пис08

Пример: Петрова О. Н., Сидоров П. А.

OK Отмена

Рисунок П2.21 – Результат разбора четвёртого элемента по полям

После ввода в структуру документа каждая ссылка становится доступной для выбора пользователем через меню «Ссылки». Прделаем «оживление» абсолютно всех ссылок на литературу, содержащихся в документе (Рисунок П2.22 – П2.24).

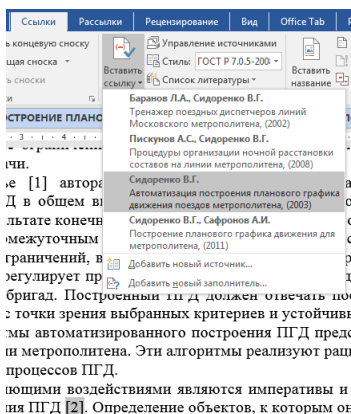


Рисунок П2.22 – «Оживление» второго элемента в тексте

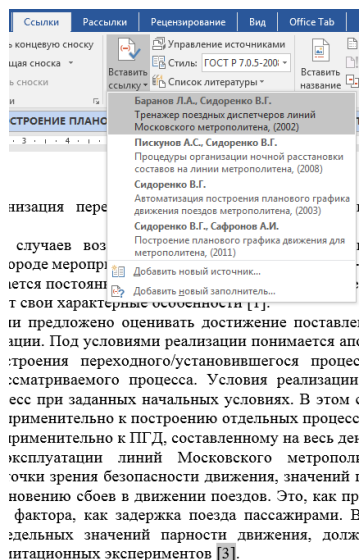


Рисунок П2.23 – «Оживление» третьего элемента в тексте

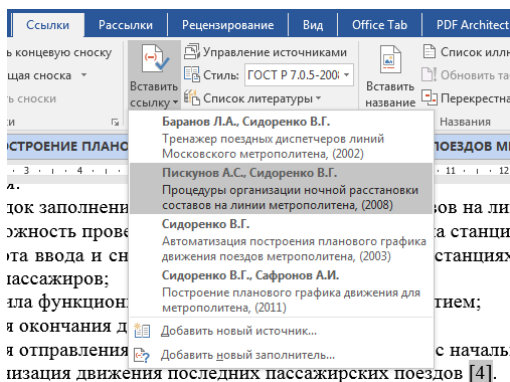


Рисунок П2.24 – «Оживление» четвёртого элемента в тексте

После «оживления» всех ссылок останется только принудительно обновить ранее размещённый блок автособираемого библиографического списка. Функция

обновления становится доступной после размещения курсора внутри блока автособираемого библиографического списка (Рисунок П2.25).

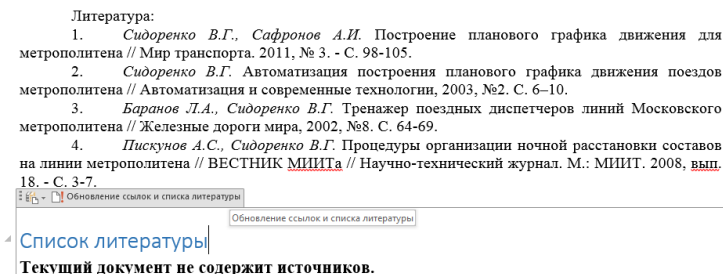


Рисунок П2.25 – Переход к обновлению автособираемого библиографического списка

Обновление приведёт к отображению только тех источников в порядке упоминания их в тексте, ссылки на которые реально размещены в тексте (Рисунок П2.26). Если источник был введён в структуру документа, но при этом не был ни разу использован в документе в виде ссылки, то он не отобразится в списке. Однако, существуют особые настройки текстового редактора, которые в любом случае позволяют выводить в конце перечня литературы, в частности, и те источники, которые не были ни разу упомянуты в тексте. Здесь таковые не рассмотрены.

Литература:

1. Сидоренко В.Г., Сафронов А.И. Построение планового графика движения для метрополитена // Мир транспорта. 2011, № 3. - С. 98-105.
2. Сидоренко В.Г. Автоматизация построения планового графика движения поездов метрополитена // Автоматизация и современные технологии, 2003, №2. С. 6–10.
3. Баранов Л.А., Сидоренко В.Г. Тренажер поездных диспетчеров линий Московского метрополитена // Железные дороги мира, 2002, №8. С. 64-69.
4. Пискунов А.С., Сидоренко В.Г. Процедуры организации ночной расстановки составов на линии метрополитена // ВЕСТНИК МИИТА // Научно-технический журнал. М.: МИИТ. 2008, вып. 18, - С. 3-7.

Обновление ссылок и списка литературы

Список литературы

1. Сидоренко В.Г. С.А.И. Построение планового графика движения для метрополитена // Мир транспорта, No. № 3, 2011. pp. С. 98-105.
2. Сидоренко В.Г. Автоматизация построения планового графика движения поездов метрополитена // Автоматизация и современные технологии, No. №2, 2003. pp. С. 6–10.
3. Баранов Л.А. С.В.Г. Тренажер поездных диспетчеров линий Московского метрополитена // Железные дороги мира, No. №8, 2002. pp. С. 64-69.
4. Пискунов А.С. С.В.Г. Процедуры организации ночной расстановки составов на линии метрополитена // ВЕСТНИК МИИТА, No. вып. 18, 2008. pp. С. 3-7]

Рисунок П2.26 – Результат создания автособираемого библиографического списка в документе.

На данном этапе можно считать поставленную задачу завершённой. И, вместе с тем, стоит отметить, что данное методическое руководство содержит достаточную базу знаний для подготовки обучающимися качественной и полной отчётной документации.

Напоследок, авторы считают нужным напомнить обучающимся о том, что по завершении любой работы перед блоком автособираемого библиографического списка они не должны забывать формулировать выводы о проделанной работе. В числе значимых пунктов раздела выводов от обучающихся требуется изложение основных положений, которые им удалось закрепить на практике благодаря выполнению работы, а также рассуждение на тему, чем работа была для них полезной, где бы в дальнейшем они смогли применить приобретённые новые знания.

Оглавление

Введение	3
1 Задание 1	5
2 Цель работы	6
3 Структура отчёта	6
4 Требования к именам файлов:	7
5 Методика оформления титульного листа:	8
6 Методика настройки нумерации страниц	14
7 Методика настройки автособираемого оглавления	17
8 Методика составления блок-схем алгоритмов программ	31
9 Примеры выполнения содержательной части задания	44
9.1. Задан фрагмент кода:	44
9.2. Задан код метода:	47
9.3. Задан фрагмент кода метода:	48
Список литературы:	50
Приложение 1. Варианты индивидуального задания:	51
Приложение 2. Методика составления автособираемого библиографического списка	63

УЧЕБНО-МЕТОДИЧЕСКОЕ ИЗДАНИЕ

Сафронов Антон Игоревич
Зольникова Надежда Николаевна
Новиков Вячеслав Геннадьевич

Составление отчётной документации по решённым задачам
алгоритмизации и программирования
Учебно-методическое пособие
для проведения аудиторных занятий по Учебной практике

Тираж 50 экз.

Изд. № 129-18

Типография ООО «ПринтСайдАп»